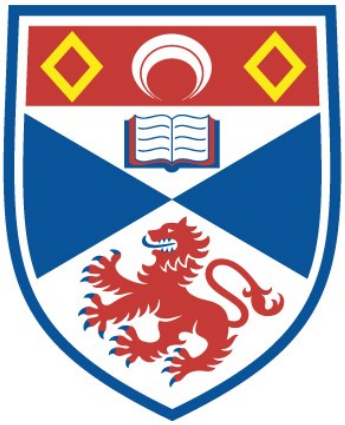
INTERVAL METHODS FOR NON-LINEAR SYSTEMS

J. M. Shearer

A Thesis Submitted for the Degree of PhD
at the
University of St Andrews



1986

Interval methods for nonlinear systems

J.M. Shearer

Thesis submitted for the degree of Doctor of Philosophy

of the University of St Andrews

ProQuest LLC.
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106 – 1346

Th VA353

Abstract

In numerical mathematics, there is a need for methods which provide a user with the solution to his problem without requiring him to understand the mathematics underlying the method of solution. Such a method involves computable tests to determine whether or not a solution exists in a given region, and whether, if it exists, such a solution may be found by using the given method. Two valuable tools for the implementation of such methods are interval mathematics and symbolic computation. In practice all computers have memories of finite size and cannot perform exact arithmetic. Therefore, in addition to the error which is inherent in a given numerical method, namely truncation error, there is also the error due to rounding. Using interval arithmetic, computable tests which guarantee the existence of a solution to a given problem in a given region, and the convergence of a particular iterative method to this solution, become practically realizable. This is not possible using real arithmetic due to the accumulation of rounding error on a computer. The advent of packages which allow symbolic computations to be carried out on a given computer is an important advance for computational numerical mathematics. In particular, the ability to compute derivatives automatically removes the need for a user to supply them, thus eliminating a major source of error in the use of methods requiring first or higher derivatives. In this thesis some methods which use interval arithmetic and symbolic computation for the solution of systems of nonlinear algebraic equations are presented.

Some algorithms based on the symmetric single-step algorithm are described. These methods however do not possess computable existence, uniqueness, and convergence tests. Algorithms which do possess such tests, based on the Krawczyk-Moore algorithm are also presented. A simple package which allows symbolic computations to be carried out is described. Several applications for such a package are given. In particular, an interval form of Brown's method is presented.

Acknowledgements

I would like to thank my supervisor Mr. M. A. Wolfe for all his help and encouragement during my time as a research student. I am grateful to the Department of Education for Northern Ireland for their financial support.

I Jonathan Marc Shearer hereby certify that this thesis which is approximately 70,000 words long has been written by me, that it is a record of work carried out by me, and that it has not been submitted in any previous application for a higher degree.

13th September 1985

I hereby certify that the candidate has fulfilled the conditions of Resolution and Regulations appropriate to the degree of Ph.D. of the University of St Andrews and that he is qualified to submit this thesis in application for that degree.

13th September 1985

I was admitted as a research student under Ordinance No. 12 on 1st October 1982 and as a candidate for the degree of Ph.D. on 1st October 1982; the higher study for which this is a record was carried out in the University of St Andrews between 1982 and 1985.

13th September 1985

Contents

1. Introduction

In numerical mathematics, there is a need for methods which provide a user with the solution to his problem without requiring him to understand the mathematics underlying the method of solution. Such a method involves computable tests to determine whether or not a solution exists in a given region, and whether, if it exists, such a solution may be found by using the given method. If these tests are not satisfied then the user must be notified of the failure of the method. If the tests are satisfied, and the problem may be solved, the user should be supplied with the solution to the problem together with rigorous bounds for the error in the computation.

Two valuable tools for the implementation of such methods are interval mathematics and symbolic computation. In practice all computers have memories of finite size and cannot perform exact arithmetic. Therefore, in addition to the error which is inherent in a given numerical method, namely truncation error, there is also the error due to rounding. Interval arithmetic began with the aim of automating the analysis of rounding and truncation error. An account is given in the thesis of Moore [Moo--62a]. Using interval arithmetic, computable tests which guarantee the existence of a solution to a given problem in a given region, and the convergence of a particular iterative method to this solution, become practically realizable. This is not possible using real arithmetic due to the accumulation of rounding error on a

computer.

The advent of packages which allow symbolic computations to be carried out on a given computer is also an important advance for computational numerical mathematics. In particular, the ability to compute derivatives automatically removes the need for a user to supply them, thus eliminating a major source of error in the use of methods requiring first or higher derivatives. Unfortunately most of the packages for symbolic computation currently available, such as MACSYMA [Bog--77a], and REDUCE 2 [Hea--73a], are difficult to interface with implementations of numerical algorithms.

In this thesis some methods for the solution of systems of nonlinear algebraic equations are presented. Nonlinear systems arise in the numerical solution of many problems in all areas of applied science. Examples of problems which give rise to nonlinear systems include optimization and the solution of ordinary and partial differential equations. Many examples are given by Ortega and Rheinboldt [OrtR-69a] and Dennis and Schnabel [DenS-83a].

The notation used throughout the thesis is described in Appendix A. In this thesis, particular emphasis will be given to methods which use interval arithmetic and symbolic computation for the solution of systems of nonlinear algebraic equations. Algorithms for estimating the solution of the system of nonlinear equations

$$f(x) = 0 \tag{1.1}$$

where $f : D \subseteq R^n \to R^n$ is a given mapping, are iterative, in that given an initial estimate $x^{(0)} \in R^n$ of the solution x^*, a sequence $\left(x^{(k)}\right)$ is generated which,

under appropriate conditions, converges to x^* . Theorem 1.1 is typical of the type of results which guarantee the existence of a solution in a given region, and the convergence of an iterative sequence to the solution.

Theorem 1.1 : (Kantorovich) Let $f : D \subseteq R^n \to R^n$ be a given mapping with $f \in C^1(\hat{D})$ where $\hat{D} \subset D$ is an open convex set. If (i) $\exists \gamma > 0$ such that

$$\|f'(x) - f'(y)\| \leq \gamma \|x - y\| \qquad (\forall x, y \in \hat{D}); \tag{1.2}$$

(ii) $\exists \hat{x} \in \hat{D}$ and $\exists \beta, \eta > 0$ such that $\left\| f'(\hat{x})^{-1} \right\| < \beta$, $\left\| f'(\hat{x})^{-1} f(\hat{x}) \right\| < \eta$, and $\alpha = \beta\eta\gamma \leq \frac{1}{2}$; (iii) $\mathrm{B}[\hat{x}, r^*] = \{x \in R^n \mid \|x - \hat{x}\| \leq r^*\} \subset \hat{D}$, where

$$r^* = \left\{1 - (1 - 2\alpha)^{\frac{1}{2}}\right\} \Big/ (\beta\gamma), \tag{1.3}$$

then (a) $\exists x^* \in \hat{D}$ such that $f(x^*) = 0$ and x^* is unique in $\mathrm{B}[\hat{x}, r^{**}] \cap \hat{D}$ where

$$r^{**} = \left\{1 + (1 - 2\alpha)^{\frac{1}{2}}\right\} \Big/ (\beta\gamma); \tag{1.4}$$

(b) the sequence $\left(x^{(k)}\right)$ generated from

$$x^{(k+1)} = x^{(k)} - f'(x^{(k)})^{-1} f(x^{(k)}) \qquad (k \geq 0) \tag{1.5}$$

with $x^{(0)} = \hat{x}$ is such that $(\forall k \geq 0)$, $x^{(k)}$ is well-defined and $x^{(k)} \in \mathrm{B}[\hat{x}, r^*]$; (c) $(\forall k \geq 0)$

$$\left\| x^* - x^{(k)} \right\| \leq (2\alpha)^{2^k} \Big/ (\beta\gamma 2^k). \quad \square \tag{1.6}$$

Theorem 1.1 is not always applicable, is not always easy to apply, and when successfully applied often yields very crude error bounds. Furthermore the error bounds are correct only when the iterates $x^{(k)}$ are computed exactly.

The presence of numerical inaccuracy due to finite computer memory and inexact arithmetic give rise to a criticism of the existence, uniqueness, and convergence theory which is typified in Theorem 1.1. In applying Theorem 1.1 it is neccessary to verify that inequalities such as those in (ii) hold. If such inequalities are to be tested by the computer, then the finite precision of the computer arithmetic must be taken into account. Therefore it is not sufficient to use machine arithmetic without allowing for rounding error. Many interval methods [Ale--72a], [Ale--77a] also suffer from the fact that the existence, uniqueness and convergence results associated with them are difficult to test using a computer.

Ideally an iterative method for estimating an isolated zero of $f : D \subseteq R^n \to R^n$ should have the following properties.

(1) There exist a computable set of sufficient conditions for the existence of a unique zero x^* of f such that

$$x_{iI}^{(0)} \leq x_i^* \leq x_{iS}^{(0)} \qquad (i = 1, \ldots, n) \tag{1.7}$$

where $x_{iI}^{(0)}$, $x_{iS}^{(0)}$ $(i = 1, \ldots, n)$ are machine numbers and are such that

$$\left\{ x \in R^n \mid x_{iI}^{(0)} \leq x_i \leq x_{iS}^{(0)} \ (i = 1, \ldots, n) \right\} \subset D.$$

(2) Using the method, the computer can generate, under the conditions referred to in (1) $2n$ sequences $\left(x_{iI}^{(k)}\right)$ and $\left(x_{iS}^{(k)}\right)$ in R such that for $i = 1, \ldots, n$

$$x_{iI}^{(k)} \leq x_{iI}^{(k+1)} \leq x_i^* \leq x_{iS}^{(k+1)} \leq x_{iS}^{(k)} \qquad (\forall k \geq 0) \tag{1.8}$$

and $\exists k^* \in N$ such that, $(\forall k \geq k^*)$ $x_{iI}^{(k)} = x_{iI}^{(k^*)}$, $x_{iS}^{(k)} = x_{iS}^{(k^*)}$, and

$$x_{iS}^{(k^*)} - x_{iI}^{(k^*)} = \varepsilon \tag{1.9}$$

where $\varepsilon > 0$ is a (small) machine-dependent real number. If the computer could have unlimited memory and could perform exact arithmetic then

$$\left| x_{iS}^{(k)} - x_{iI}^{(k)} \right| \to 0 \qquad (k \to \infty). \tag{1.10}$$

(3) There exist a set of **computable** sufficient conditions for the non-existence of a zero x^* of f, such that (1.7) holds; these conditions contain entities which would, in any case, be computed in the implementation of the method.

Property (1) allows us to establish the existence of a zero x^* of f in a given subset of D using the computer, and Property (2) then guarantess that the sequences $\left(x_I^{(k)}\right)$ and $\left(x_S^{(k)}\right)$ will converge to x^* from above and below respectively. Since the sequences $\left(x_I^{(k)}\right)$ and $\left(x_S^{(k)}\right)$ are actually computed by the machine, they provide, by virtue of (1.8), rigorous error bounds with rounding taken into account. Property (2) also contains a natural stopping criterion. By (1.8), for $i = 1, \ldots, n$, $x_{iS}^{(k)} - x_{iI}^{(k)}$ decreases monotonically as k increases $(\forall k < k^*)$ and then remains constant $(\forall k \geq k^*)$. The precision of the machine arithmetic sets a natural bound ε on the width of the interval $\left[x_{iI}^{(k)}, x_{iS}^{(k)}\right]$ which is guaranteed to contain x_i^* $(i = 1, \ldots, n)$. Numerical convergence can therefore be considered to have been obtained when $x_{iI}^{(k+1)} = x_{iI}^{(k)}$ and $x_{iS}^{(k+1)} = x_{iS}^{(k)}$ $(i = 1, \ldots, n)$, for some k.

It would be possible, using a method which has properties (1)–(3), to analyze a so-called box $\hat{D}$ where

$$\hat{D} = \left\{ x \in D \mid \hat{x}_{iI} \leq x_i \leq \hat{x}_{iS} \ (i = 1, \ldots, n) \right\},$$

by systematically dividing $\hat{D}$ into sub-boxes, rejecting those sub-boxes which Property (3) indicates do not contain a zero of f, and constructing a list of sub-boxes of $\hat{D}$ which Property (1) indicates contain a unique solution of f which may be bounded by Property (2). Several iterative methods which have properties (1)–(3) now exist.

Moore [Moo--66a] appears to have been the originator of the first interval arithmetic form of Newton's method for the solution of systems of nonlinear algebraic equations with the introduction of the interval Newton operator $\underline{\mathbf{N}} : I(R^n) \to I(R^n)$ defined by

$$\underline{\mathbf{N}}(\underline{\mathbf{x}}) = \mathrm{m}(\underline{\mathbf{x}}) - \underline{\mathbf{f}}'(\underline{\mathbf{x}})^{-1} f(\mathrm{m}(x)). \tag{1.11}$$

Hansen [Han--68a] has modified Moore's method by using a technique for obtaining a narrower inclusion of the Jacobian of f. Both of these methods however require the *inversion* of an interval matrix. The Gauss algorithm for interval matrix inversion is applicable only when the interval matrix is, for example, strictly diagonally dominant, and pivoting techniques [Heb--74a] are not in general effective. Kupermann and Hansen [AleH-83a], [Kup--67a], [Han--69a] have described a strategy for transforming the matrix to become strictly diagonally dominant. Several authors, among whom are Alefeld and Herzberger [AleH-83a], and Monsch [Mon--73a], have presented techniques for bounding the inverse but there is still no robust, generally applicable algorithm for the inversion of an interval matrix.

Krawczyk [Kra--69a] has removed the need to invert interval matrices by introducing the Krawczyk operator $\underline{\mathbf{K}} : I(R^n) \to I(R^n)$ defined by

$$\underline{\mathbf{K}}(\underline{\mathbf{x}}) = y - g(y) + \underline{\mathbf{R}}(\underline{\mathbf{x}})(\underline{\mathbf{x}} - y) \tag{1.12}$$

where

$$g(y) = Yf(y),$$

$$\underline{R}(\underline{x}) = I - Y\underline{f}'(\underline{x}),$$

$y \in \underline{x}$ is arbitrary and $Y \in M(R^n)$ is any non-singular point matrix. Krawczyk [Kra--69a] and Nickel [Nic--71a] appear to have been the first authors to give satisfactory existence and convergence theorems for the Krawczyk and interval Newton methods respectively. Alefeld and Herzberger [AleH-72a] have also developed a form of the interval Newton method, which has been improved by Madsen [Mad--73a] but Madsen's methods requires an interval matrix containing the inverse of the Jacobian to be known.

Moore [Moo--77a] has made a major advance by developing simple computationally verifiable sufficient conditions for the existence of a unique zero of f in a given box, and the convergence of a modified Krawczyk algorithm to this zero. This modified algorithm has become known as the Krawczyk-Moore algorithm. The computational effectiveness of the Krawczyk-Moore algorithm has been greatly enhanced by the search procedure of Moore and Jones [Jon--78a], [MooJ-77a], [Jon--80a] by means of which a given box can be systematically sub-divided into smaller boxes in order to find a safe box $\hat{\underline{x}}$, which Moore's computationally verifiable tests [Moo--77a] show to contain a unique zero of f to which the Krawczyk-Moore algorithm with $\underline{x}^{(0)} = \hat{\underline{x}}$ will converge. A further advance has been made by Moore [Moo--78a] who has shown that the computationally verifiable conditions corresponding to the Krawczyk-Moore algorithm can be simplified when $\underline{x}^{(0)} \in I(R^n)$ is an n-cube. In the same paper, Moore has shown that the sequence $\left(y^{(k)}\right)$ generated from the modified Newton method

$$y^{(k+1)} = y^{(k)} - Yf(y^{(k)}) \qquad (k \geq 0) \tag{1.13}$$

with $y^{(0)} \in \underline{x}^{(0)}$ arbitrary and $Y \approx \{m(\underline{f}'(\underline{x}^{(0)}))\}^{-1}$ converges to x^*, the unique zero of f in $\underline{x}^{(0)}$. This appears to be the first published result of its kind. Qi [Qi---80a] has generalized Moore's results to boxes other than n-cubes by introducing an alternative norm.

An article reviewing the "state of the art" in the application of interval analysis to the solution of systems of nonlinear algebraic equations has been written by Moore [Moo--78b]. Moore [Moo--80a] gives a list of the principal non-existence, existence, uniqueness and convergence results which were known in 1979/80. A more detailed account of the Moore-Jones search procedure is also given in this paper. The computational effectiveness of the existence and uniqueness results due to Moore [Moo--78a] has been clearly demonstated by Rall [Ral--80a] who has shown that the computational labour required to apply the Kantorovich theorem (Theorem 1.1) is far greater than that which is required to apply Moore's theorem.

A major source of computational labour in the Krawczyk-Moore algorithm arises from the need to compute $I - Y\underline{f}'(\underline{x}^{(k)})$ $(k \geq 0)$. Wolfe [Wol--80a] has introduced a modification of the Krawczyk-Moore algorithm in which the value of $\underline{f}'$ and Y are re-used for several *inner iterations*, thereby significantly reducing the computational labour involved.

An important modification of the Krawczyk-Moore algorithm has been made by Hansen and Sengupta [HanS-81a] who have replaced $\underline{K} : I(R^n) \to I(R^n)$ with $\underline{H} : I(R^n) \to I(R^n)$ defined by

$$\underline{H}_i(\underline{x}) = x_i - g_i(x) + \sum_{j=1}^{i-1} \underline{R}_{ij}(\underline{x})(\underline{H}_j'(\underline{x}) - x_j) + \sum_{j=i}^{n} \underline{R}_{ij}(\underline{x})(\underline{x}_j - x_j), \quad (1.14)$$

$$\underline{\mathrm{H}}_i'(\underline{\mathrm{x}}) = \underline{\mathrm{H}}_i(\underline{\mathrm{x}}) \cap \underline{\mathrm{x}}_i \qquad (i = 1, \ldots, n), \tag{1.15}$$

where $x_i = \mathrm{m}(\underline{\mathrm{x}}_i)$ $(i = 1, \ldots, n)$. Moore and Qi [MooQ-82a] have described comprehensive existence, uniqueness and convergence results for the modified algorithm.

Hansen [Han--78a] has introduced an extended interval arithmetic in which the inverse of an interval containing zero can be computed, and has applied these ideas to the interval Newton method. Hansen and Sengupta [HanS-81a] have shown how these ideas can be used together with their modification of the Krawczyk-Moore algorithm to give a more efficient algorithm. Hansen and Greenberg [HanG-83a] have used this algorithm as the basis for a hybrid interval Newton method.

Alefeld and Platzöder [AleP-83a] have introduced a method similar to the Krawczyk-Moore algorithm, but requiring much less computational labour in each iteration. Qi [Qi---82a] has suggested a further improvement to the Krawczyk-Moore algorithm and has introduced an existence-convergence result involving the second derivative of f , which appears to be the only result of its kind.

There are many applications of symbolic computation in numerical mathematics. See for example [Ral--80b] and [How--79a]. Of particular interest is the work on symbolic manipulation involving factorable functions [Ral--69a], [Pug--72a], [McC--83a]. Sisser has shown [Sis--82a] how interval extensions of factorable functions may be generated by a computer. Sisser [Sis--82b], [Sis--82c] has also shown how symbolic computation may be used to invert an interval Hessian matrix and to improve Newton's method for nonlinear minimization.

2. Preliminary results

This chapter contains certain results which are used subsequently. Much other background material is contained in [OrtR-69a], [AleH-83a], and [Var--62a].

Lemma 2.1 : Let $A \in M(R^n)$. Then given any $\varepsilon > 0$, there is a norm $\|\cdot\| : M(R^n) \to R$ such that

$$\|A\| \leq \rho(A) + \varepsilon.$$

Proof : A proof of Lemma 2.1 is given by Ortega and Rheinboldt ([OrtR-69a], 2.2.8). □

Lemma 2.2 : Let $A \in M(R^n)$ and assume that $A \geq 0$. Then $(I - A)^{-1}$ exists and is nonnegative if and only if $\rho(A) < 1$.

Proof : A proof of Lemma 2.2 is given by Ortega and Rheinboldt ([OrtR-69a], 2.4.5). □

Definition 2.1 : A matrix $A \in M(R^n)$ is an M-matrix if and only if A is invertible, $A^{-1} \geq 0$, and $a_{ij} \leq 0 \quad (i, j = 1, \ldots, n;\ i \neq j)$. □

Lemma 2.3 : Let $A \in M(R^n)$ and $B \in M(R^n)$. If $|B| \leq A$, then $\rho(B) \leq \rho(A)$.

Proof : A proof of Lemma 2.3 is given by Ortega and Rheinboldt ([OrtR-69a], 2.4.9). □

Lemma 2.4 : Let $A_1 \in M(R^n)$ be an M-matrix with diagonal part D_1 , and off-diagonal part $-B_1 = A_1 - D_1$. If $D_2 \in M(R^n)$ is any nonnegative diagonal matrix and $B_2 \in M(R^n)$ is any nonnegative matrix with zero diagonal satisfying $B_2 \leq B_1$, then $A = (D_1 + D_2) - (B_1 - B_2)$ is an M-matrix and $A^{-1} \leq {A_1}^{-1}$.

Proof : A proof of Lemma 2.4 is given by Ortega and Rheinboldt ([OrtR-69a], 2.4.10). □

Definition 2.2 : Let P , Q , R be real $n \times n$ matrices. Then

$$P = Q - R$$

is a regular splitting of P if and only if $\exists\, Q^{-1} \geq 0$ and $R \geq 0$. □

Lemma 2.5 : If $P = Q - R$ is a regular splitting of the real $n \times n$ matrix P and $P^{-1} \geq 0$ then

$$\rho\left(Q^{-1}R\right) < 1.$$

Proof : A proof of this result is given by Varga ([Var--62a] Theorem 3.13). □

Definition 2.3 : Let $g : R^n \to R^n$ be a given mapping. Then g is a P-contraction if and only if $\exists P \in M(R^n)$ such that $P \geq 0$, $|g(x) - g(y)| < P|x - y|$ $(\forall x, y \in R^n)$ and $\rho(P) < 1$. □

Lemma 2.6 : If $g : R^n \to R^n$ is a P-contraction then g has a unique fixed point $x^* \in R^n$. Furthermore if $x^{(0)} \in R^n$ and the sequence $\left(x^{(k)}\right)$ is generated from

$$x^{(k+1)} = g(x^{(k)}) \qquad (k \geq 0)$$

then $x^{(k)} \to x^*$ $(k \to \infty)$.

Proof : A proof of Lemma 2.6 is given by Ortega and Rheinboldt ([OrtR-69a], 13.1.2). □

Lemma 2.7 : Let $h : R^n \times R^n \to R^n$ be a given mapping and suppose that $\exists P, Q \in M(R^n)$ such that $P \geq 0$, $\rho(P) < 1$, $Q \geq 0$, $\rho(Q) < 1$. Suppose

also that

$$|h(\hat{u},v)-h(\tilde{u},v)| \leq P|\hat{u}-\tilde{u}| \qquad (\forall \hat{u},\tilde{u},v\in R^n),$$
$$|h(u,\hat{v})-h(u,\tilde{v})| \leq Q|\hat{v}-\tilde{v}| \qquad (\forall u,\hat{v},\tilde{v}\in R^n),$$

and that the sequence $\left(x^{(k)}\right)$ is generated from

$$y^{(k)} = h(y^{(k)}, x^{(k)}),$$
$$x^{(k+1)} = h(y^{(k)}, x^{(k+1)}) \qquad (k\geq 0),$$

with $x^{(0)}\in R^n$ given. If $\rho(S)<1$ where

$$S = (I-Q)^{-1}P(I-P)^{-1}Q$$

then h has a unique fixed point $x^*\in R^n$ and $x^{(k)}\to x^*$ $(k\to\infty)$.

Proof : That $\left(x^{(k)}\right)$ is defined follows from Lemma 2.6 because $(\forall \hat{x}\in R^n)$ $h(\cdot,\hat{x}) : R^n \to R^n$ is a P-contraction and $(\forall \hat{y}\in R^n)$ $h(\hat{y},\cdot) : R^n\to R^n$ is a Q-contraction. If $\hat{x},\tilde{x}\in R^n$ are given and $\hat{y}=h(\hat{y},\hat{x})$, $\hat{z}=h(\hat{y},\hat{z})$, $\tilde{y}=h(\tilde{y},\tilde{x})$, $\tilde{z}=h(\tilde{y},\tilde{z})$, then by Lemma 2.2

$$|\tilde{y}-\hat{y}| \leq (I-P)^{-1}Q|\tilde{x}-\hat{x}|$$

and

$$|\tilde{z}-\hat{z}| \leq (I-Q)^{-1}P|\tilde{y}-\hat{y}|.$$

So if $g : R^n\to R^n$ is defined by

$$y = h(y,x),$$
$$g(x) = h(y,g(x)),$$

then g is an S-contraction and $x^{(k+1)} = g(x^{(k)})$ $(\forall k\geq 0)$. So by Lemma 2.6, g has a unique fixed point $x^*\in R^n$ and $x^{(k)}\to x^*$ $(k\to\infty)$. Finally it is easy to show that

$$x^* = g(x^*) \iff x^* = h(x^*,x^*). \qquad \square$$

Lemma 2.8 : Let $h : R^n \times R^n \times R^n \to R^n$ be a given mapping and suppose that $\exists P, Q, R \in M(R^n)$ such that $P \geq 0$, $\rho(P) < 1$, $Q \geq 0$, $\rho(Q) < 1$, $R \geq 0$, $\rho(R) < 1$. Suppose also that

$$\begin{aligned} |h(\hat{u}, v, w) - h(\tilde{u}, v, w)| &\leq P\,|\hat{u} - \tilde{u}| && (\forall \hat{u}, \tilde{u}, v, w \in R^n), \\ |h(u, \hat{v}, w) - h(u, \tilde{v}, w)| &\leq Q\,|\hat{v} - \tilde{v}| && (\forall u, \hat{v}, \tilde{v}, w \in R^n), \\ |h(u, v, \hat{w}) - h(u, v, \tilde{w})| &\leq R\,|\hat{w} - \tilde{w}| \,. && (\forall u, v, \hat{w}, \tilde{w} \in R^n), \end{aligned}$$

and that the sequence $\left(x^{(k)}\right)$ is generated from

$$\begin{aligned} y^{(k)} &= h\left(y^{(k)}, x^{(k)}, x^{(k)}\right) \\ x^{(k+1)} &= h\left(y^{(k)}, x^{(k+1)}, y^{(k)}\right) \qquad (\forall k \geq 0) \end{aligned}$$

with $x^{(0)} \in R^n$ given. If $\rho(S) < 1$ where

$$S = (I - Q)^{-1}(P + R)(I - P)^{-1}(Q + R)$$

then h has a unique fixed point $x^* \in R^n$ and $x^{(k)} \to x^* \quad (k \to \infty)$.

Proof : That $\left(x^{(k)}\right)$ is defined follows from Lemma 2.6 because $(\forall \hat{x} \in R^n)$ $h(\cdot, \hat{x}, \hat{x}) : R^n \to R^n$ is a P-contraction and $(\forall \hat{y} \in R^n)$. $h(\hat{y}, \cdot, \hat{y}) : R^n \to R^n$ is a Q-contraction. If $\hat{x}, \tilde{x} \in R^n$ are given and $\hat{y} = h(\hat{y}, \hat{x}, \hat{x})$, $\hat{z} = h(\hat{y}, \hat{z}, \hat{y})$, $\tilde{y} = h(\tilde{y}, \tilde{x}, \tilde{x})$ and $\tilde{z} = h(\tilde{y}, \tilde{z}, \tilde{y})$ then by Lemma 2.2

$$|\tilde{y} - \hat{y}| \leq (I - P)^{-1}(Q + R)\,|\tilde{x} - \hat{x}|\,,$$

and

$$|\tilde{z} - \hat{z}| \leq (I - Q)^{-1}(P + R)\,|\tilde{y} - \hat{y}|\,.$$

So if $g : R^n \to R^n$ is defined by

$$\begin{aligned} y &= h(y, x, x) \\ g(x) &= h(y, g(x), y) \end{aligned}$$

then g is an S-contraction and $x^{(k+1)} = g(x^{(k)})$ $(\forall k \geq 0)$. So by Lemma 2.6, g has a unique fixed point $x^* \in R^n$ and $x^{(k)} \to x^*$ $(k \to \infty)$.

If $x^* = g(x^*)$ then since

$$y^* = h(y^*, x^*, x^*)$$

and

$$x^* = h(y^*, x^*, y^*)$$

it follows that

$$|y^* - x^*| \leq R\,|x^* - y^*|$$

whence by Lemma 2.2 $x^* = y^*$ and so $x^* = h(x^*, x^*, x^*)$. Conversely, if $x^* = h(x^*, x^*, x^*)$, then

$$|y^* - x^*| \leq P\,|y^* - x^*|$$

whence $x^* = y^*$ and so

$$\begin{aligned} |g(x^*) - x^*| &= |h(x^*, g(x^*), x^*) - h(x^*, x^*, x^*)| \\ &\leq Q\,|g(x^*) - x^*)| \end{aligned}$$

whence $g(x^*) = x^*$. □

Lemma 2.9 : (Taylor's Theorem) If $f : D \subseteq R^n \to R^n$ is a given mapping with $f \in C^1(\hat{D})$ where $\hat{D} \subset D$ is an open convex set then $(\forall x, y \in \hat{D})$ $\exists \theta_i \in [0,1]$ $(i = 1, \ldots, n)$ such that for $i = 1, \ldots, n$,

$$f_i(y) - f_i(x) = \sum_{j=1}^{n} \partial_j f_i(\xi_i)(y_j - x_j)$$

where $\xi_i = (x + \theta_i(y - x))$.

Proof : A proof of Lemma 2.9 is given by Ortega and Rheinboldt ([OrtR-69a], 3.2.2). □

Lemma 2.10 : Let $f : S \subseteq R^n \to R^n$ be G-differentiable in an open neighbourhood $\widehat{S} \subseteq S$ of a point x^* at which $f' : \widehat{S} \to M(R^n)$ is continuous and $f(x^*) = 0$. Suppose that $f'(x) = P(x) - Q(x) \quad (\forall x \in \widehat{S})$ where $P : \widehat{S} \to M(R^n)$ is continuous in $\widehat{S}$, $P(x^*)$ is nonsingular, and $\rho(H(x^*)) < 1$ where $H(x) = P(x)^{-1} Q(x)$. Then there exists an open neighbourhood S^* of x^* such that for any $x^{(0)} \in S^*$ and any sequence (m_k) of positive integers, the sequence $(x^{(k)})$ generated from

$$x^{(k+1)} = x^{(k)} - \left(H(x^{(k)})^{m_k - 1} + \cdots + I\right) P(x^{(k)})^{-1} f(x^{(k)}) \qquad (k \geq 0)$$

is defined and converges to x^* . Furthermore, if $m_k \to \infty \quad (k \to \infty)$ then $(x^{(k)})$ converges R-superlinearly.

Proof : A proof of Lemma 2.10 is given by Ortega and Rheinboldt ([OrtR-69a], 10.3.1). □

Definition 2.4 : The sequence $(\underline{x}^{(k)})$ in $I(R^n)$ is **nested** if and only if $\underline{x}^{(k+1)} \subseteq \underline{x}^{(k)} \quad (\forall k \geq 0)$. □

Definition 2.5 : Let $\underline{f} : I(D) \subset I(R^n) \to I(R)$ be a given mapping. Then $\underline{f}$ is **inclusion monotonic** if and only if $(\underline{x} \subseteq \underline{y} \in I(D) \quad \Rightarrow \quad \underline{f}(\underline{x}) \subseteq \underline{f}(\underline{y}))$. □

Lemma 2.11 : If $\left(\underline{\mathbf{x}}^{(k)}\right)$ is a nested sequence in $I(R^n)$ then $\exists \underline{\mathbf{x}}^* \in I(R^n)$ such that $\underline{\mathbf{x}}^* \subseteq \underline{\mathbf{x}}^{(k)} \quad (\forall k \geq 0)$ and $\underline{\mathbf{x}}^{(k)} \to \underline{\mathbf{x}}^* \quad (k \to \infty)$.

Proof : By hypothesis we have, for $i = 1, \ldots, n$

$$x_{iI}^{(0)} \leq x_{iI}^{(k)} \leq x_{iI}^{(k+1)} \leq x_{iS}^{(k+1)} \leq x_{iS}^{(k)} \leq x_{iS}^{(0)} \qquad (\forall k \geq 0).$$

Therefore, for $i = 1, \ldots, n$ the sequence $\left(x_{iI}^{(k)}\right)$ is monotonic increasing and is bounded above by $x_{iS}^{(0)}$. Therefore $\exists x_{iI}^* \leq x_{iS}^{(0)}$ such that $x_{iI}^{(k)} \uparrow x_{iI}^* \quad (k \to \infty)$. Similarly $\exists x_{iS}^* \geq x_{iI}^{(0)}$ such that $x_{iS}^{(k)} \downarrow x_{iS}^* \quad (k \to \infty)$. If for some i , $x_{iI}^* > x_{iS}^*$ it is easily shown that $\exists \widehat{k} \geq 0$ such that $x_{iI}^{(k)} > x_{iS}^{(k)} \quad (\forall k \geq \widehat{k})$ which contradicts the hypothesis that $\underline{\mathbf{x}}^{(k)} \in I(R^n) \quad (\forall k \geq 0)$. Therefore $x_{iI}^* \leq x_{iS}^* \quad (i = 1, \ldots, n)$ and the result follows. □

Lemma 2.12 : (Brouwer's Theorem) Every continuous mapping $f : D \subseteq R^n \to R^n$, where D is a convex, bounded, closed subset of R^n , which maps D into itself has at least one fixed point $u \in D$; that is

$$(\forall x \in D,\ f(x) \in D) \quad \Rightarrow \quad (\exists u \in D \text{ such that } u = f(u)).$$

Proof : A proof of Lemma 2.12 is given by Brouwer [Bro--12a]. □

Lemma 2.13 : Let $f : D \subseteq R^n \to R^n$ be given with $f \in C^1(\widehat{D})$ where $\widehat{D} \subset D$ is an open convex set. Let $\underline{\mathbf{f}}' : I(\widehat{D}) \to I(M(R^n))$ be an inclusion monotonic interval

extension of $f' : \hat{D} \to M(R^n)$. The Krawczyk operator $\underline{\mathbb{K}} : I(\hat{D}) \to I(R^n)$ is defined by

$$\underline{\mathbb{K}}(\underline{x}) = \mathrm{m}(\underline{x}) - Y f(\mathrm{m}(\underline{x})) + (I - Y\underline{f}'(\underline{x}))(\underline{x} - \mathrm{m}(\underline{x})),$$

where $Y \in M(R^n)$ is nonsingular. If $\underline{\mathbb{K}}(\hat{\underline{x}}) \subseteq \hat{\underline{x}}$, then $\exists x^* \in \underline{\mathbb{K}}(\hat{\underline{x}})$ such that $f(x^*) = 0$ and the sequence $\left(x^{(k)}\right)$ generated from

$$x^{(k+1)} = x^{(k)} - Y f(x^{(k)}) \qquad (k \geq 0)$$

with $x^{(0)} = \mathrm{m}(\hat{\underline{x}})$, converges to x^* . Furthermore if $\underline{\mathbb{K}}(\hat{\underline{x}}) \subset \mathrm{int}(\hat{\underline{x}})$ then x^* is the unique zero of f in $\hat{\underline{x}}$.

Proof : A proof of Lemma 2.13 is given by Moore and Qi [MooQ-82a]. □

3. Methods for the solution of a class of nonlinear algebraic equations based on the symmetric single-step algorithm

Let $f : R^n \to R^n$ be defined by

$$f(x) = Ax + Td(x) + c \tag{3.1}$$

in which A and T are real $n \times n$ matrices with

$$\begin{aligned} A &= D - L - U \\ &= D - B, \end{aligned} \tag{3.2}$$

and

$$\begin{aligned} T &= T_D - T_L - T_U \\ &= T_D - T_B \end{aligned} \tag{3.3}$$

where D and T_D are diagonal, with positive diagonal elements, $L \geq 0$ and $T_L \leq 0$ are strictly lower triangular, $U \geq 0$ and $T_U \leq 0$ are strictly upper triangular, $c \in R^n$, $d : R^n \to R^n$ is continuous, diagonal and isotone, and is G-differentiable in an open subset of R^n . The equation $f(x) = 0$ arises from discretizations of various boundary-value problems [Hen--62a], [Kel--68a], [Rhe--74a], and its solution is therefore of considerable importance.

Alefeld [Ale--72a] has obtained results for the relaxed total-step method (RG) and relaxed single-step method (RE) for the solution of $f(x) = 0$ with $T = I$ and has also described some generalizations (RGI, REI, RGID, REID, REIDK) in which interval arithmetic is used. Alefeld has also obtained some results for the generalized linear iterative methods NRG and NRE in which the system of linear algebraic equations corresponding to Newton's Method is solved by using the relaxed total-step and relaxed single-step methods respectively. He has described some generalizations (NRGID, NREID, NREIDK) in which interval arithmetic is used and he also describes modifications (NRGID*, NREID*, NREIDK*) which under appropriate conditions are superlinearly convergent. Additional information on these and other algorithms is given in [AleH-83a].

Alefeld [Ale--77a] has also considered the so-called symmetric single-step method (SE, SEDK) for the solution of systems of linear algebraic equations with interval coefficients.

Some real methods and some interval methods for the solution of $f(x) = 0$ which are based on the symmetric single-step method are described in this chapter. Computational experience indicates that these methods can be more efficient than the corresponding methods of Alefeld [Ale--72a] which are based upon the single-step method.

3.1 Preliminaries

The following Lemmas are used subsequently.

Lemma 3.1 : If $f : R^n \to R^n$ is given by (3.1)–(3.3) then

$$\left| (D + T_D d)^{-1}(x) - (D + T_D d)^{-1}(y) \right| \leq D^{-1} |x - y| .$$

Proof : Let $r : R^n \to R^n$ be defined by

$$\begin{aligned} r(x) &= (D + T_D d)(x) \\ &= Dx + T_D d(x). \end{aligned}$$

Now for $i = 1, \ldots, n$

$$\begin{aligned} |u_i - v_i| &\leq \left| u_i - v_i + \frac{t_{ii}}{a_{ii}} (d_i(u_i) - d_i(v_i)) \right| \\ &= \frac{1}{a_{ii}} |\{a_{ii} u_i + t_{ii} d_i(u_i)\} - \{a_{ii} v_i + t_{ii} d_i(v_i)\}| . \end{aligned}$$

So

$$|u - v| \leq D^{-1} |r(u) - r(v)| .$$

Let $x = r(u)$, $y = r(v)$. Then

$$\left| r^{-1}(x) - r^{-1}(y) \right| \leq D^{-1} |x - y|$$

whence the result follows. □

Lemma 3.2 : If $f : R^n \to R^n$ is given by (3.1)–(3.3), $\exists \mu > 0$ such that $(\forall x, y \in R^n)$ $|d(x) - d(y)| \leq \mu |x - y|$, and $\mu |t_{ij}| < |a_{ij}|$ $(i, j = 1, \ldots, n)$ then $(\forall x, y \in R^n)$

$$|B(x - y) + T_B(d(x) - d(y))| \leq B |x - y| .$$

Proof : By the isotonicity of d we have $(\forall x, y \in R^n)$

$$|(-a_{ij})(x_j - y_j) + (-t_{ij})(d_j(x_j) - d_j(y_j))| \leq |a_{ij}|\,|x_j - y_j| \quad . \ (j \neq i)$$

whence the result follows. □

Lemma 3.3 : Let $A \in M(R^n)$ be given by (3.2). Then

$$\left(\rho\left((D-U)^{-1}L(D-L)^{-1}U\right) < 1\right) \iff \left(\rho\left(D^{-1}B\right) < 1\right).$$

Proof : Let $M = D^{-1}L$, $N = D^{-1}U$. Then Lemma 2.2 gives

$$(I-M)^{-1} \geq 0$$

and

$$(I-N)^{-1} \geq 0$$

whence

$$(I-N)^{-1}\,(I-M)^{-1}\,MN \geq 0.$$

Now

$$\begin{aligned}(D-U)^{-1}L(D-L)^{-1}U &= (I-N)^{-1}M(I-M)^{-1}N \\ &= (I-N)^{-1}\,(I-M)^{-1}\,MN.\end{aligned}$$

Therefore by Lemma 2.2 since $\rho\left((D-U)^{-1}L(D-L)^{-1}U\right) < 1$ it follows that

$$\begin{aligned}0 &\leq \left\{I - (I-N)^{-1}\,(I-M)^{-1}\,MN\right\}^{-1} \\ &= \{I-(M+N)\}^{-1}(I-M)(I-N).\end{aligned}$$

Post-multiplying by $\left((I-N)^{-1}\,(I-M)^{-1}\right)$ gives

$$0 \leq \{I - (M+N)\}^{-1}$$

whence by Lemma 2.2 $\rho(M+N) < 1$. But $M+N = D^{-1}L + D^{-1}U = D^{-1}B$. Therefore $\rho\left(D^{-1}B\right) < 1$.

Conversely suppose that

$$1 > \rho\left(D^{-1}B\right) = \rho(M+N).$$

Since $M+N \geq 0$, by Lemma 2.2

$$(I - (M+N))^{-1} \geq 0.$$

Also

$$I - (M+N) = (I-M)(I-N) - MN$$

is a regular splitting of $(I - (M+N))$. Therefore by Lemma 2.5

$$1 > \rho\left(\{(I-M)(I-N)\}^{-1} MN\right)$$
$$= \rho\left((D-U)^{-1} L (D-L)^{-1} U\right). \square$$

3.2 The algorithms SSS1 and MSSS1

The most natural application of the symmetric single-step method to the solution of $f(x) = 0$ where $f : R^n \to R^n$ is defined by (3.1) results in the algorithm **SSS1**, which consists of generating the sequence $\left(x^{(k)}\right)$ by solving

$$Dy^{(k)} + T_D d(y^{(k)}) - Ly^{(k)} - T_L d(y^{(k)}) - Ux^{(k)} - T_U d(x^{(k)}) + c = 0 \quad (3.4)$$

for $y^{(k)}$ and then solving

$$Dx^{(k+1)} + T_D d(x^{(k+1)}) - Ly^{(k)} - T_L d(y^{(k)}) - Ux^{(k+1)} - T_U d(x^{(k+1)}) + c = 0 \quad (3.5)$$

for $x^{(k+1)}$. We have the following result.

Theorem 3.1 : If (i) $f : R^n \to R^n$ is defined by (3.1) ; (ii) $\exists \mu > 0$ such that $(\forall x, y \in R^n) \quad |d(x) - d(y)| \leq \mu |x - y|$; (iii) $\mu |t_{ij}| < |a_{ij}| \quad (i, j = 1, \ldots, n)$ where $T = (t_{ij})_{n \times n}$, $A = (a_{ij})_{n \times n}$; (iv) $\rho(S) < 1$ where

$$S = (D - U)^{-1} L (D - L)^{-1} U,$$

then the SSS1 sequence $\left(x^{(k)}\right)$, generated from (3.4) – (3.5) with $x^{(0)} \in R^n$ arbitrary, converges to the unique solution of $f(x) = 0$ in R^n .

Proof : Since $D \geq 0$ and $T_D \geq 0$ are diagonal and invertible, and $d : R^n \to R^n$ is continuous, diagonal and isotone, $(D + T_D d)^{-1} : R^n \to R^n$ exists. Let $h : R^n \times R^n \to R^n$ be defined by

$$h(u, v) = (D + T_D d)^{-1} \{Lu + T_L d(u) + Uv + T_U d(v) - c\}.$$

Then the SSS1 sequence $\left(x^{(k)}\right)$ is generated from

$$y^{(k)} = h(y^{(k)}, x^{(k)}),$$
$$x^{(k+1)} = h(y^{(k)}, x^{(k+1)}) \qquad (\forall k \geq 0).$$

From Lemma 3.1 and (iii) we have, for $i = 1, \ldots, n$,

$$\begin{aligned}
\left(|h(\hat{u}, v) - h(\tilde{u}, v)|\right)_i &= \Big(\big| (D + T_D d)^{-1} (L\hat{u} + T_L d(\hat{u}) + Uv + T_U d(v) - c) \\
&\qquad - (D + T_D d)^{-1} (L\tilde{u} + T_L d(\tilde{u}) + Uv + T_U d(v) - c) \big| \Big)_i \\
&\leq \left(D^{-1} \, |(L\hat{u} + T_L d(\hat{u})) - (L\tilde{u} + T_L d(\tilde{u}))| \right)_i \\
&\leq \left(D^{-1} L \, |\hat{u} - \tilde{u}| \right)_i,
\end{aligned}$$

whence

$$|h(\hat{u}, v) - h(\tilde{u}, v)| \leq P \, |\hat{u} - \tilde{u}|$$

and similarly

$$|h(u,\hat{v}) - h(u,\tilde{v})| \leq Q\,|\hat{v} - \tilde{v}|$$

where $P = D^{-1}L$, $Q = D^{-1}U$, $P \geq 0$; $\rho(P) = 0 < 1$, $Q \geq 0$ and $\rho(Q) = 0 < 1$. Furthermore if

$$S = (I - Q)^{-1}P(I - P)^{-1}Q$$

then

$$S = (D - U)^{-1}L(D - L)^{-1}U$$

and $\rho(S) < 1$. The remainder of the proof is a direct consequence of Lemma 2.7. □

A considerable saving in computational labour may be made by noting that $(\forall k \geq 0)$ $x_n^{(k+1)} = y_n^{(k)}$ and $y_1^{(k+1)} = x_1^{(k+1)}$. Computational experience shows that a little more labour may be saved by setting $y_1^{(0)} = x_1^{(0)}$ but Theorem 3.1 is not then applicable. The modification of SSS1 in which $y_1^{(k)} = x_1^{(k)}$ and $x_n^{(k+1)} = y_n^{(k)}$ $(\forall k \geq 0)$ will be referred to as MSSS1.

3.3 The algorithms SSS2 and MSSS2

In SSS1, (3.4) and (3.5) are equivalent to solving $2n$ nonlinear algebraic equations in one real variable, each of which must be solved using a subsidiary iterative algorithm. The algorithm SSS2 is an attempt to avoid the problem of solving the $2n$ equations and consists of generating the sequence $(x^{(k)})$ by solving the linear equation

$$Dy^{(k)} + T_D d(x^{(k)}) - Ly^{(k)} - T_L d(y^{(k)}) - Ux^{(k)} - T_U d(x^{(k)}) + c = 0 \qquad (3.6)$$

for $y^{(k)}$ and then solving the linear equation

$$Dx^{(k+1)} + T_D d(y^{(k)}) - Ly^{(k)} - T_L d(y^{(k)}) - Ux^{(k+1)} - T_U d(x^{(k+1)}) + c = 0 \quad (3.7)$$

for $x^{(k+1)}$. The following theorem shows that the sequence $\left(x^{(k)}\right)$ generated from SSS2 converges to the unique solution of $f(x) = 0$ under hypotheses which are often valid when the equation $f(x) = 0$ arises from the discretization of boundary value problems.

Theorem 3.2 : If (i) $f : R^n \to R^n$ is defined by (3.1) ; (ii) $\exists \mu > 0$ such that $(\forall x, y \in R^n) \quad |d(x) - d(y)| \leq \mu |x - y|$; (iii) $\mu |t_{ij}| < |a_{ij}| \quad (i, j = 1, \ldots, n)$ where $T = (t_{ij})_{n \times n}$, $A = (a_{ij})_{n \times n}$; (iv) $\rho(S) < 1$ where

$$S = (D - U)^{-1} (L + \mu T_D) (D - L)^{-1} (U + \mu T_D),$$

then the sequence $\left(x^{(k)}\right)$, generated from SSS2 with $x^{(0)} \in R^n$ arbitrary, converges to the unique solution of $f(x) = 0$ in R^n .

Proof : Let $h : R^n \times R^n \times R^n \to R^n$ be defined by

$$h(u, v, w) = D^{-1} \{Lu + T_L d(u) + Uv + T_U d(v) - T_D d(w) - c\}$$

Then the SSS2 sequence $\left(x^{(k)}\right)$ is generated from

$$y^{(k)} = h(y^{(k)}, x^{(k)}, x^{(k)}),$$

$$x^{(k+1)} = h(y^{(k)}, x^{(k+1)}, y^{(k)}) \qquad (\forall k \geq 0).$$

On setting $P = D^{-1} L$, $Q = D^{-1} U$, and $R = \mu D^{-1} T_D$, the rest of the proof follows immediately from Lemma 2.8. □

It follows from (3.6) and (3.7) that $y_n^{(k)}$ differs from $x_n^{(k+1)}$ in one term only, and $x_1^{(k+1)}$ differs from $y_1^{(k+1)}$ in one term only. This leads to the conjecture that it might be possible to save computational labour without significantly

reducing the rate of convergence of the SSS2 sequence by replacing $x_n^{(k+1)}$ with $y_n^{(k)}$ and $y_1^{(k)}$ with $x_1^{(k)}$, thereby obtaining the algorithm MSSS2 which consists of generating $\left(x^{(k)}\right)$ from

$$y_1^{(k)} = x_1^{(k)},$$

$$y_i^{(k)} = a_{ii}^{-1}\Big\{ - t_{ii}d_i(x_i^{(k)}) - \sum_{j=1}^{i-1} a_{ij}y_j^{(k)} - \sum_{j=1}^{i-1} t_{ij}d_j(y_j^{(k)}) - \sum_{j=i+1}^{n} a_{ij}x_j^{(k)} - \sum_{j=i+1}^{n} t_{ij}d_j(x_j^{(k)}) - c_i\Big\} \qquad (i = 2, \ldots, n), \tag{3.8}$$

$$x_n^{(k+1)} = y_n^{(k)},$$

$$x_i^{(k+1)} = a_{ii}^{-1}\Big\{ - t_{ii}d_i(y_i^{(k)}) - \sum_{j=1}^{i-1} a_{ij}y_j^{(k)} - \sum_{j=1}^{i-1} t_{ij}d_j(y_j^{(k)}) - \sum_{j=i+1}^{n} a_{ij}x_j^{(k+1)} - \sum_{j=i+1}^{n} t_{ij}d_j(x_j^{(k+1)}) - c_i\Big\} \qquad (i = n-1, \ldots, 1), \tag{3.9}$$

The following theorem shows that the sequence $\left(x^{(k)}\right)$ generated from MSSS2 converges to the unique solution of $f(x) = 0$, where $f : R^n \to R^n$ is defined by (3.1), under hypotheses which are often valid when the equation $f(x) = 0$ arises from the discretization of boundary value problems.

Theorem 3.3 : If (i) $f : R^n \to R^n$ is defined by (3.1) ; (ii) $\exists \mu > 0$ such that $(\forall x, y \in R^n) \quad |d(x) - d(y)| \leq \mu |x - y|$; (iii) $\mu |t_{ij}| < |a_{ij}| \quad (i, j = 1, \ldots, n)$ where $T = (t_{ij})_{n\times n}$, $A = (a_{ij})_{n\times n}$; (iv) $\rho\left(D^{-1}B\right) < 1$, then f has a unique zero $x^* \in R^n$. Furthermore if $\gamma = \left\| D^{-1}(B + \mu T_D)\right\| < 1$ then the sequence $\left(x^{(k)}\right)$ generated from MSSS2 with $x^{(0)} \in R^n$ arbitrary converges to x^* and $\left\|x^{(k+1)} - x^*\right\| \leq \gamma \left\|x^{(k)} - x^*\right\| \quad (\forall k \geq 0)$.

Proof : Because D and T_D are diagonal with positive diagonal elements, and $d : R^n \to R^n$ is continuous diagonal and isotone, it follows that $(D + T_D d)^{-1} : R^n \to R^n$ exists. Let $g : R^n \to R^n$ be defined by

$$g(x) = (D + T_D d)^{-1} (Bx + T_B d(x) - c).$$

Then $(x = g(x)) \iff (f(x) = 0)$. Furthermore by Lemma 3.1 and Lemma 3.2 $(\forall x, y \in R^n)$

$$\left| (D + T_D d)^{-1} (x) - (D + T_D d)^{-1} (y) \right| \leq D^{-1} |x - y|$$

and

$$|B(x - y) + T_B(d(x) - d(y))| \leq B |x - y|,$$

whence $(\forall x, y \in R^n)$ $|g(x) - g(y)| \leq D^{-1} B |x - y|$. Therefore g is a $(D^{-1} B)$ -contraction and by Lemma 2.6 f has a unique zero $x^* \in R^n$. Now

$$\gamma = \max_{1 \leq i \leq n} \left\{ |a_{ii}{}^{-1} t_{ii} \mu| + \sum_{j \neq i} |a_{ii}{}^{-1} a_{ij}| \right\} < 1$$

so by (3.8) , $\left| y_1^{(k)} - x_1^* \right| = \left| x_1^{(k)} - x_1^* \right|$ and $\left| y_i^{(k)} - x_i^* \right| \leq \gamma \left\| x^{(k)} - x^* \right\|$ $(i = 2, \ldots, n)$. So by (3.9) $\left| x_n^{(k+1)} - x_n^* \right| \leq \gamma \left\| x^{(k)} - x^* \right\|$ and $\left\| x_i^{(k+1)} - x_i^* \right\| \leq \gamma \left\| x^{(k)} - x^* \right\|$ $(i = n - 1, \ldots, 1)$. Therefore $\left\| x^{(k+1)} - x^* \right\| \leq \gamma \left\| x^{(k)} - x^* \right\|$. □

3.4 The algorithms NSSS1 and NMSSS1

The algorithm which is obtained by using a variable number m_k of iterations of the symmetric single-step method to solve the system of linear algebraic equations corresponding to the k^{th} iteration in Newton's method will be called the Newton

Symmetric Single-step method (NSSS1); it belongs to the class of generalized linear iterative methods [OrtR-69a].

It may be shown that, if $f : R^n \to R^n$ is a given G-differentiable mapping and NSSS1 is applied to the equation $f(x) = 0$ to obtain the sequence $\left(x^{(k)}\right)$ then

$$x^{(k+1)} = x^{(k)} - \left(H^{(k)^{m_k-1}} + \cdots + H^{(k)} + I\right) P^{(k)^{-1}} f\left(x^{(k)}\right)$$

where

$$\begin{aligned} H^{(k)} &= P^{(k)^{-1}} Q^{(k)} \\ &= \left(D^{(k)} - U^{(k)}\right)^{-1} D^{(k)} \left(D^{(k)} - L^{(k)}\right)^{-1} L^{(k)} D^{(k)^{-1}} U^{(k)} \\ &= \left(D^{(k)} - U^{(k)}\right)^{-1} L^{(k)} \left(D^{(k)} - L^{(k)}\right)^{-1} U^{(k)}, \end{aligned}$$

in which

$$\begin{aligned} P^{(k)} &= \left(D^{(k)} - L^{(k)}\right) D^{(k)^{-1}} \left(D^{(k)} - U^{(k)}\right), \\ Q^{(k)} &= L^{(k)} D^{(k)^{-1}} U^{(k)}, \end{aligned}$$

and

$$f'\left(x^{(k)}\right) = D^{(k)} - L^{(k)} - U^{(k)}, \tag{3.10}$$

where $D^{(k)}$ is diagonal, $L^{(k)}$ is strictly lower triangular, and $U^{(k)}$ is strictly upper triangular. We have the following theorem.

Theorem 3.4 : If (i) $f : S \subseteq R^n \to R^n$ is G-differentiable in an open neighbourhood $\widehat{S} \subseteq S$ of a point $x^* \in S$ such that $f(x^*) = 0$ and $f' : \widehat{S} \to M(R^n)$ is continuous at x^* ; (ii) $f'(x^*) = F_D - F_L - F_U$ where F_D is diagonal and nonsingular, F_L is strictly lower triangular, and F_U is strictly upper triangular;

(iii) $\rho(H^*) < 1$ where $H^* = (F_D - F_U)^{-1} F_L (F_D - F_L)^{-1} F_U$, then there is an open neighbourhood S^* of x^* such that $S^* \subset S$ and for any $x^{(0)} \in S^*$ and any sequence (m_k) the NSSS1 sequence $(x^{(k)})$ is defined, and $x^{(k)} \to x^*$ $(k \to \infty)$. Furthermore, if $m_k \to \infty$ $(k \to \infty)$ then $(x^{(k)})$ converges R-superlinearly.

Proof : In Lemma 2.10 let $P : S^* \to M(R^n)$ and $Q : S^* \to M(R^n)$ be defined by

$$P(x) = (D(x) - L(x))\, D(x)^{-1}\, (D(x) - U(x))$$

and

$$Q(x) = L(x)\, D(x)^{-1}\, U(x)$$

where $D : S^* \to M(R^n)$, $L : S^* \to M(R^n)$ and $U : S^* \to M(R^n)$ are such that $(\forall x \in S^*)$, $D(x)$ is diagonal, $L(x)$ is strictly lower triangular, $U(x)$ is strictly upper triangular, and

$$f'(x) = D(x) - L(x) - U(x).$$

Then $(\forall x \in S^*)$

$$f'(x) = P(x) - Q(x)$$

and

$$\begin{aligned} H^* &= H(x^*) \\ &= P(x^*)^{-1} Q(x^*). \end{aligned}$$

The result which it is required to prove now follows from Lemma 2.10. □

Corollary 3.1 : Suppose that in (3.1), A is an M-matrix. Let $S \subseteq R^n$ be an open convex set, and let $\mu > 0$ be such that $(\forall x \in S) \quad 0 < d_i'(x) \leq \mu$ $(i = 1, \ldots, n)$. Suppose that $\mu |t_{ij}| \leq |a_{ij}| \quad (i, j = 1, \ldots, n \; ; \; i \neq j)$ and that $\rho\left((D-U)^{-1} L (D-L)^{-1} U\right) < 1$. If $f : R^n \to R^n$ defined by (3.1) has a zero $x^* \in S$ then there is an open neighbourhood S^* of x^* such that $S^* \subset S$ and for any $x^{(0)} \in S^*$ and any sequence (m_k) the NSSS1 sequence $(x^{(k)})$ is defined and converges to x^*. Furthermore, if $m_k \to \infty \quad (k \to \infty)$ then $(x^{(k)})$ converges R-superlinearly.

Proof : Since $0 < d_i'(x^*) \leq \mu \quad (i = 1, \ldots, n)$ and $\mu |t_{ij}| \leq |a_{ij}| \quad (i, j = 1, \ldots, n \; ; \; i \neq j)$, it follows that

$$0 \leq L + T_L d'(x^*) \leq L$$

and

$$0 \leq U + T_U d'(x^*) \leq U.$$

Let $A_1 = D - U$. Then because $\rho\left(D^{-1} U\right) = 0 < 1$, it follows from Lemma 2.2 that

$$A_1{}^{-1} = (I - D^{-1} U)^{-1} D^{-1} \geq 0.$$

So A_1 is an M-matrix. Let $D_1 = D$, $B_1 = U$, $D_2 = T_D d'(x^*)$, and $B_2 = -T_U d'(x^*)$. Then $D_2 \geq 0$, $B_2 \geq 0$, and if

$$\begin{aligned} A_2 &= D_1 + D_2 - (B_1 - B_2) \\ &= D + T_D d'(x^*) - U - T_U d'(x^*) \end{aligned}$$

then by Lemma 2.4 A_2 is an M-matrix and $A_2{}^{-1} \leq A_1{}^{-1}$. Therefore

$$0 \leq (D + T_D d'(x^*) - U - T_U d'(x^*))^{-1} \leq (D - U)^{-1}.$$

Similarly

$$0 \leq (D + T_D d'(x^*) - L - T_L d'(x^*))^{-1} \leq (D - L)^{-1}.$$

So if $F_D = D + T_D d'(x^*)$, $F_L = L + T_L d'(x^*)$, $F_U = U + T_U d'(x^*)$ and $H^* = (F_D - F_U)^{-1} F_L (F_D - F_L)^{-1} F_U$, then

$$0 \leq H^* \leq (D - U)^{-1} L (D - L)^{-1} U.$$

Therefore by Lemma 2.3

$$\rho(H^*) \leq \rho\left((D - U)^{-1} L (D - L)^{-1} U\right) < 1.$$

The results which are to be proved now follow from Theorem 3.4. □

The hypothesis $\rho\left((D - U)^{-1} L (D - L)^{-1} U\right) < 1$ may be replaced with $\rho\left(D^{-1} B\right) < 1$ in both Theorem 3.1 and Corollary 3.1. This result follows from Lemma 3.3.

In the implementation of NSSS1, the symmetric single-step method is used to obtain an approximate solution of the system of linear algebraic equations

$$f'(x^{(k)})x = f'(x^{(k)})x^{(k)} - f(x^{(k)}).$$

If $x^{(k,m)}$ $(m = 0, \ldots, m_k)$ are the iterates generated from the symmetric single-step algorithm then $x^{(k,0)} = x^{(k)}$ and $x^{(k+1)} = x^{(k,m_k)}$. If $A^{(k)} = f'(x^{(k)}) = D^{(k)} - L^{(k)} - U^{(k)}$, where $D^{(k)}$ is diagonal, $L^{(k)}$ is strictly lower triangular, and $U^{(k)}$ is strictly upper triangular, and $b^{(k)} = f'(x^{(k)})x^{(k)} - f(x^{(k)})$ then the $x^{(k,m)}$ are computed from

$$y^{(k,m)} = D^{(k)^{-1}} L^{(k)} y^{(k,m)} + D^{(k)^{-1}} U^{(k)} x^{(k,m)} + D^{(k)^{-1}} b^{(k)},$$

$$x^{(k,m+1)} = D^{(k)^{-1}} L^{(k)} y^{(k,m)} + D^{(k)^{-1}} U^{(k)} x^{(k,m+1)} + D^{(k)^{-1}} b^{(k)}.$$

A significant saving in computational labour may be made by noting that $(\forall m \geq 0)$ $x_n^{(k,m+1)} = y_n^{(k,m)}$ and $y_1^{(k,m+1)} = x_1^{(k,m+1)}$. Computational experience shows that it is possible to save a little more labour by setting $y_1^{(k,0)} = x_1^{(k,0)}$ also, but Theorem 3.4 is not then applicable. The modification of NSSS1 in which $y_1^{(k,m)} = x_1^{(k,m)}$ and $x_n^{(k,m+1)} = y_n^{(k,m)}$ $(m = 0, \ldots, m_k - 1)$ $(\forall k \geq 0)$ will be called NMSSS1.

3.5 The algorithms ISSS1 and IMSSS1

The most natural application of an interval form of the symmetric single-step method for the solution of $f(x) = 0$ where $f : R^n \to R^n$ is given by (3.1) involves the solution of several nonlinear algebraic equations in one variable for each iteration, in much the same way as for SSS1. The algorithm ISSS1 is a modification of the "most natural" application in the same spirit as SSS2 and consists of generating the sequence $(\underline{x}^{(k)})$ from

$$\underline{y}_i^{(k)} = \underline{h}_i(\underline{y}^{(k)}, \underline{x}^{(k)}, \underline{x}^{(k)}) \cap \underline{x}_i^{(k)} \qquad (i = 1, \ldots, n), \tag{3.11}$$

$$\underline{x}_i^{(k+1)} = \underline{h}_i(\underline{y}^{(k)}, \underline{x}^{(k+1)}, \underline{y}^{(k)}) \cap \underline{y}_i^{(k)} \qquad (i = n, \ldots, 1), \tag{3.12}$$

where $\underline{\mathbf{h}} : I(R^n) \times I(R^n) \times I(R^n) \to I(R^n)$ is defined by

$$\underline{\mathbf{h}}(\underline{\mathbf{u}}, \underline{\mathbf{v}}, \underline{\mathbf{w}}) = D^{-1} \{L\underline{\mathbf{u}} + T_L \underline{\mathbf{d}}(\underline{\mathbf{u}}) + U\underline{\mathbf{v}} + T_U \underline{\mathbf{d}}(\underline{\mathbf{v}}) - T_D \underline{\mathbf{d}}(\underline{\mathbf{w}}) - c\}. \tag{3.13}$$

The following theorem is valid.

Theorem 3.5 : If (i) $f : R^n \to R^n$ is defined by (3.1) and (3.2)–(3.3) hold; (ii) $d : R^n \to R^n$ is continuous and diagonal; (iii) $\underline{\mathbf{d}} : I(R^n) \to I(R^n)$ is a

continuous inclusion monotonic interval extension of d ; (iv) $\exists \nu > 0$ such that $(\forall \underline{x} \in I(\underline{x}^{(0)}))$ $\mathrm{w}(\underline{d}(\underline{x})) \leq \nu \mathrm{w}(\underline{x})$; (v) $\rho(|D^{-1}|(|B| + \nu|T|)) < 1$; (vi) $\exists x^* \in \underline{x}^{(0)}$ such that $f(x^*) = 0$, then the sequence $(\underline{x}^{(k)})$ generated from ISSS1 is defined and $x^* \in \underline{x}^{(k+1)} \subseteq \underline{x}^{(k)}$ $(\forall k \geq 0)$. Furthermore $\underline{x}^{(k)} \to x^*$ $(k \to \infty)$.

Proof : That $x^* \in \underline{x}^{(k+1)} \subseteq \underline{x}^{(k)}$ $(\forall k \geq 0)$ follows from (3.11) – (3.13) by a simple inductive argument. Because $\underline{x}^{(k+1)} \subseteq \underline{x}^{(k)}$ $(\forall k \geq 0)$, by Lemma 2.11 $\exists \underline{x}^* \in I(R^n)$ such that $\underline{x}^* \subseteq \underline{x}^{(k)}$ $(\forall k \geq 0)$, and $\underline{x}^{(k)} \to \underline{x}^*$ $(k \to \infty)$. Furthermore by the continuity of $\underline{h}$ and of the intersection mapping

$$\mathrm{w}(\underline{x}^*) \leq \mathrm{w}(\underline{h}(\underline{x}^*, \underline{x}^*, \underline{x}^*)) \leq |D^{-1}|(|B| + \nu|T|)\mathrm{w}(\underline{x}^*),$$

whence by (v), $\mathrm{w}(\underline{x}^*) = 0$. Therefore $\underline{x}^{(k)} \to x^*$ $(k \to \infty)$. □

As in the real case, it follows from (3.11) and (3.12) that $\underline{y}_n^{(k)}$ differs from $\underline{x}_n^{(k+1)}$ in one term only. This suggests a modification of the ISSS1 sequence in which $\underline{x}_n^{(k+1)}$ is replaced by $\underline{y}_n^{(k)}$ and $\underline{x}_1^{(k+1)}$ is replaced by $\underline{y}_1^{(k+1)}$, thereby obtaining the algorithm IMSSS1 which consists of generating $(\underline{x}^{(k)})$ from

$$\begin{aligned}
\underline{y}_1^{(k)} &= \underline{x}_1^{(k)}, \\
\underline{y}_i^{(k)} &= \underline{h}_i(\underline{y}^{(k)}, \underline{x}^{(k)}, \underline{x}^{(k)}) \cap \underline{x}_i^{(k)} \qquad (i = 2, \ldots, n), \\
\underline{x}_n^{(k+1)} &= \underline{y}_n^{(k)}, \\
\underline{x}_i^{(k+1)} &= \underline{h}_i(\underline{y}^{(k)}, \underline{x}^{(k+1)}, \underline{y}^{(k)}) \cap \underline{y}_i^{(k)} \qquad (i = n-1, \ldots, 1).
\end{aligned}$$

The proof of the following theorem is similar to that of Theorem 3.5.

Theorem 3.6 : If the hypotheses of Theorem 3.5 are valid then the sequence

$\left(\underline{x}^{(k)}\right)$ generated from IMSSS1 is defined and $x^* \in \underline{x}^{(k+1)} \subseteq \underline{x}^{(k)} \quad (\forall k \geq 0)$. Furthermore $\underline{x}^{(k)} \to x^* \quad (k \to \infty)$. □

3.6 The algorithms INSSS1 and INMSSS1

Let $f : S \subseteq R^n \to R^n$ be a given mapping and suppose that $f \in C^1(S)$. Let $\hat{\underline{x}} \in I(S)$ be given and suppose that f has exactly one zero $x^* \in \hat{\underline{x}}$. If $\hat{x} \in \hat{\underline{x}}$ then by Lemma 2.9 $\exists \theta_i \in [0,1] \quad (i = 1, \ldots, n)$ such that

$$J(x^*, \hat{x})x^* = J(x^*, \hat{x})\hat{x} - f(\hat{x}) \tag{3.14}$$

where

$$J(x^*, \hat{x}) = \left(\hat{J}_{ij}\right)_{n\times n}$$
$$= \left(\partial_j f_i(\hat{x} + \theta_i(x^* - \hat{x}))\right)_{n\times n} .$$

For $i = 1, \ldots, n$, if $\hat{J}_{ii} \neq 0$ then by (3.14)

$$x_i^* = \hat{x}_i - \hat{J}_{ii}^{-1}\left\{\sum_{j=1}^{i-1} \hat{J}_{ij}(x_j^* - \hat{x}_j) + \sum_{j=i+1}^{n} \hat{J}_{ij}(x_j^* - \hat{x}_j) + f_i(\hat{x})\right\}.$$

Let $\underline{f}' : I(S) \to I(M(R^n))$ be a continuous inclusion monotonic interval extension of $f' : S \to M(R^n)$ where $f'(x) = \left(\partial_j f_i(x)\right)_{n\times n} \quad (x \in S)$. Then $J(x^*, \hat{x}) \in \underline{f}'(\hat{\underline{x}})$. Therefore if $\left(\underline{x}^{(k)}\right)$ is generated from the interval Newton symmetric single-step algorithm (INSSS1)

$$\underline{x}^{(k,0)} := \underline{x}^{(k)} \tag{3.15}$$

$$x^{(k,m)} = \mathrm{m}\left(\underline{x}^{(k,m)}\right) \tag{3.16}$$

$$\underline{u}_i^{(k,m)} = x_i^{(k,m)} - \underline{F}_{ii}^{(k)^{-1}}\left\{\sum_{j=1}^{i-1} \underline{F}_{ij}^{(k)}\left(\underline{y}_j^{(k,m)} - x_j^{(k,m)}\right)\right.$$

$$+ \sum_{j=i+1}^{n} \underline{F}_{ij}^{(k)} (\underline{x}_j^{(k,m)} - x_j^{(k,m)}) + f_i(x^{(k,m)})\}, \tag{3.17}$$

$$\underline{y}_i^{(k,m)} = \underline{u}_i^{(k,m)} \cap \underline{x}_i^{(k,m)} \quad (i = 1, \ldots, n) \tag{3.18}$$

$$\underline{v}_i^{(k,m)} = x_i^{(k,m)} - \underline{F}_{ii}^{(k)^{-1}} \{ \sum_{j=1}^{i-1} \underline{F}_{ij}^{(k)} (\underline{y}_j^{(k,m)} - x_j^{(k,m)})$$

$$+ \sum_{j=i+1}^{n} \underline{F}_{ij}^{(k)} (\underline{x}_j^{(k,m+1)} - x_j^{(k,m)}) + f_i(x^{(k,m)})\}, \tag{3.19}$$

$$\underline{x}_i^{(k,m+1)} = \underline{v}_i^{(k,m)} \cap \underline{y}_i^{(k,m)} \quad (i = n, \ldots, 1) \quad (m = 0, \ldots, m_k - 1) \tag{3.20}$$

$$\underline{x}^{(k+1)} = \underline{x}^{(k,m_k)} \quad (\forall k \geq 0) \tag{3.21}$$

with $\underline{x}^{(0)} \subseteq \hat{\underline{x}}$, where $\underline{F}_{ij}^{(k)} = \partial_j \underline{f}_i(\underline{x}^{(k)})$ $(i, j = 1, \ldots, n)$ and if $0 \notin \underline{F}_{ii}^{(0)}$ $(i = 1, \ldots, n)$ then $(\forall k \geq 0)$ $\underline{x}^{(k)}$ is defined and $x^* \in \underline{x}^{(k+1)} \subseteq \underline{x}^{(k)}$.

Theorem 3.7 : Let $f : S \subseteq R^n \to R^n$ and $\hat{\underline{x}} \in I(S)$ be given. If (i) $f \in C^1(S)$; (ii) $\underline{f}' : I(S) \to I(M(R^n))$ is a continuous inclusion monotonic interval extension of $f' : S \to M(R^n)$; (iii) $(\forall \underline{x} \in I(S))$ $\underline{f}'(\underline{x}) = \underline{D}(\underline{x}) - \underline{L}(\underline{x}) - \underline{U}(\underline{x})$ where $\underline{D} : I(S) \to I(M(R^n))$, $\underline{L} : I(S) \to I(M(R^n))$, and $\underline{U} : I(S) \to I(M(R^n))$ are such that $(\forall \underline{x} \in I(S))$, $\underline{D}(\underline{x})$ is diagonal, $\underline{L}(\underline{x})$ is strictly lower triangular, and $\underline{U}(\underline{x})$ is strictly upper triangular; (iv) $0 \notin \underline{D}(\hat{\underline{x}})$; (v) f has exactly one zero $x^* \in \hat{\underline{x}}$; (vi) the sequence $(\underline{x}^{(k)})$ is generated from **INSSS1** with $\underline{x}^{(0)} = \hat{\underline{x}}$, then $\underline{x}^{(k)}$ is defined $(\forall k \geq 0)$, $\exists \underline{x}^* \in I(R^n)$ such that $x^* \in \underline{x}^* \subseteq \underline{x}^{(k+1)} \subseteq \underline{x}^{(k)}$ $(\forall k \geq 0)$ and $\underline{x}^{(k)} \to \underline{x}^*$ $(k \to \infty)$.

Proof : That $\underline{x}^{(k)}$ is defined and $\underline{x}^{(k+1)} \subseteq \underline{x}^{(k)}$ $(\forall k \geq 0)$ follows from a simple inductive argument using (3.15)–(3.21), (iv), and the inclusion monotonicity of $\underline{f}'$.

That $x^* \in \underline{x}^{(k)}$ $(\forall k \geq 0)$ follows from a simple inductive argument based on (3.14) and (3.15)–(3.21). That $\exists \underline{x}^*$ such that $x^* \in \underline{x}^*$ and $\underline{x}^{(k)} \to \underline{x}^*$ $(k \to \infty)$ follows from Lemma 2.11 since $\underline{x}^{(k+1)} \subseteq \underline{x}^{(k)}$ $(\forall k \geq 0)$. □

Theorem 3.8 : If the hypotheses of Theorem 3.7 are valid and if also $\rho\left(|\underline{D}(\hat{\underline{x}})|^{-1} |\underline{B}(\hat{\underline{x}})|\right) < 1$ where $\underline{B}(\underline{x}) = \underline{L}(\underline{x}) + \underline{U}(\underline{x})$ $(\underline{x} \in I(S))$ then $\underline{x}^{(k)} \to x^*$ $(k \to \infty)$.

Proof : By the continuity of the mappings (3.16)–(3.20) it follows that $\exists \underline{x}^*, \underline{u}^* \in I(S)$ such that for $i = 1, \ldots, n$

$$\underline{u}_i^* = \mathrm{m}(\underline{x}_i^*) - \underline{F}_{ii}^{*-1} \{ \sum_{j \neq i} \underline{F}_{ij}^* (\underline{x}_j^* - \mathrm{m}(\underline{x}_j^*)) + f_i(\mathrm{m}(\underline{x}^*)) \}$$

and $\underline{x}_i^* = \underline{u}_i^* \cap \underline{x}_i^*$ where $\underline{F}^* = \underline{f}'(\underline{x}^*)$. Therefore $\underline{x}^* \subseteq \underline{u}^*$, and because $0 \notin \underline{D}(\underline{x}^*)$ it follows by a similar argument to one which was used by Alefeld [Ale--72a] that for $i = 1, \ldots, n$,

$$0 \in \sum_{j \neq i} \underline{F}_{ij}^* (\underline{x}_j^* - \mathrm{m}(\underline{x}_j^*)) + f_i(\mathrm{m}(\underline{x}^*)).$$

Therefore, for $i = 1, \ldots, n$,

$$\begin{aligned} \mathrm{w}(\underline{x}_i^*) &\leq \mathrm{w}(\underline{u}_i^*) \\ &= \left| \underline{F}_{ii}^{*-1} \right| \sum_{j \neq i} |\underline{F}_{ij}^*| \, \mathrm{w}(\underline{x}_j^*), \end{aligned}$$

whence

$$\left(I - \left| \underline{D}^{*-1} \right| |\underline{B}^*| \right) \mathrm{w}(\underline{x}^*) \leq 0, \tag{3.22}$$

where $\underline{\mathbf{D}}^* = \underline{\mathbf{D}}(\underline{\mathbf{x}}^*)$ and $\underline{\mathbf{B}}^* = \underline{\mathbf{B}}(\underline{\mathbf{x}}^*)$. Now $\underline{\mathbf{x}}^* \subseteq \hat{\underline{\mathbf{x}}}$ so $\underline{\mathbf{D}}^* \subseteq \underline{\mathbf{D}}(\hat{\underline{\mathbf{x}}})$ and $\underline{\mathbf{B}}^* \subseteq \underline{\mathbf{B}}(\hat{\underline{\mathbf{x}}})$. Therefore $\left|\underline{\mathbf{D}}^{*-1}\right|\left|\underline{\mathbf{B}}^*\right| \leq \left|\underline{\mathbf{D}}(\hat{\underline{\mathbf{x}}})^{-1}\right|\left|\underline{\mathbf{B}}(\hat{\underline{\mathbf{x}}})\right|$, whence by Lemma 2.3 $\rho\left(\left|\underline{\mathbf{D}}^{*-1}\right|\left|\underline{\mathbf{B}}^*\right|\right) < 1$ and so by Lemma 2.2 $\left(I - \left|\underline{\mathbf{D}}^{*-1}\right|\left|\underline{\mathbf{B}}^*\right|\right)^{-1} \geq 0$. Therefore by (3.22), $\mathrm{w}(\underline{\mathbf{x}}^*) = 0$, whence $\underline{\mathbf{x}}^{(k)} \to x^* \quad (k \to \infty)$. □

Theorem 3.9 : If the hypotheses of Theorem 3.8 are valid and if also $m_k \to \infty$ $(k \to \infty)$ then $(\forall k \geq 0) \quad \exists c^{(k)} \geq 0$ such that $\left\|\mathrm{w}\left(\underline{\mathbf{x}}^{(k+1)}\right)\right\| \leq c^{(k)} \left\|\mathrm{w}\left(\underline{\mathbf{x}}^{(k)}\right)\right\|$ where $\|\cdot\| : R^n \to R$ is any multiplicative norm, and $c^{(k)} \to 0 \quad (k \to \infty)$.

Proof : For $i = 1, \ldots, n$

$$\mathrm{w}\left(\underline{\mathbf{x}}_i^{(k,m+1)}\right) \leq \mathrm{w}\left(\underline{\mathbf{v}}_i^{(k,m)}\right)$$
$$\leq \left|\underline{\mathbf{F}}_{ii}^{(k)-1}\right| \sum_{j \neq i} \left|\underline{\mathbf{F}}_{ij}^{(k)}\right| \mathrm{w}\left(\underline{\mathbf{x}}_j^{(k,m)}\right) + \mathrm{w}\left(\underline{\mathbf{F}}_{ii}^{(k)-1}\right) \left|f_i\left(\underline{\mathbf{x}}^{(k,m)}\right)\right|,$$

where $\underline{\mathbf{F}}^{(k)} = \underline{\mathbf{f}}'\left(\underline{\mathbf{x}}^{(k)}\right)$. So

$$\mathrm{w}\left(\underline{\mathbf{x}}^{(k,m+1)}\right) \leq \left|\underline{\mathbf{D}}^{(k)-1}\right| \left|\underline{\mathbf{B}}^{(k)}\right| \mathrm{w}\left(\underline{\mathbf{x}}^{(k,m)}\right) + \mathrm{w}\left(\underline{\mathbf{D}}^{(k)-1}\right) \left|f(x^{(k,m)}) - f(x^*)\right|$$

where $\underline{\mathbf{D}}^{(k)} = \underline{\mathbf{D}}(\underline{\mathbf{x}}^{(k)})$, and $\underline{\mathbf{B}}^{(k)} = \underline{\mathbf{B}}(\underline{\mathbf{x}}^{(k)}) = \underline{\mathbf{L}}(\underline{\mathbf{x}}^{(k)}) + \underline{\mathbf{U}}(\underline{\mathbf{x}}^{(k)})$. By Lemma 2.9

$$\left|f(x^{(k,m)}) - f(x^*)\right| \leq \left|\underline{\mathbf{f}}'\left(\underline{\mathbf{x}}^{(k,m)}\right)\right| \left|x^{(k,m)} - x^*\right|$$
$$\leq \frac{1}{2} \left|\underline{\mathbf{F}}^{(k)}\right| \mathrm{w}\left(\underline{\mathbf{x}}^{(k,m)}\right).$$

Therefore

$$\mathrm{w}\left(\underline{\mathbf{x}}^{(k+1)}\right) \leq \left(M^{(k)}\right)^{m_k} \mathrm{w}\left(\underline{\mathbf{x}}^{(k)}\right) \qquad (\forall k \geq 0), \tag{3.23}$$

where

$$M^{(k)} = \left|\underline{\mathbf{D}}^{(k)^{-1}}\right|\left|\underline{\mathbf{B}}^{(k)}\right| + \frac{1}{2}\mathrm{w}\left(\underline{\mathbf{D}}^{(k)^{-1}}\right)\left|\underline{\mathbf{F}}^{(k)}\right|.$$

Now by the inclusion monotonicity of $\underline{\mathbf{f}}'$, $\left|\underline{\mathbf{D}}^{(k+1)^{-1}}\right|\left|\underline{\mathbf{B}}^{(k+1)}\right| \leq \left|\underline{\mathbf{D}}^{(k)^{-1}}\right|\left|\underline{\mathbf{B}}^{(k)}\right|$ $(\forall k \geq 0)$. Furthermore, since $\underline{\mathbf{x}}^{(k)} \to x^*$ $(k \to \infty)$, $\mathrm{w}\left(\underline{\mathbf{D}}^{(k)^{-1}}\right)\left|\underline{\mathbf{F}}^{(k)}\right| \to 0$ $(k \to \infty)$. Therefore if $\alpha^{(k)} = \rho\left(M^{(k)}\right)$ $(\forall k \geq 0)$ then $\alpha^{(k)} \downarrow \rho\left(|D^{*^{-1}}||B^*|\right)$ $(k \to \infty)$ where $D^* = \underline{\mathbf{D}}(x^*)$ and $B^* = \underline{\mathbf{B}}(x^*)$. But $x^* \in \hat{\underline{\mathbf{x}}}$ so $0 \leq |D^{*^{-1}}||B^*| \leq \left|\underline{\mathbf{D}}(\hat{\underline{\mathbf{x}}})^{-1}\right||\underline{\mathbf{B}}(\hat{\underline{\mathbf{x}}})|$ whence $\rho\left(|D^{*^{-1}}||B^*|\right) \leq \rho\left(\left|\underline{\mathbf{D}}(\hat{\underline{\mathbf{x}}})^{-1}\right||\underline{\mathbf{B}}(\hat{\underline{\mathbf{x}}})|\right) < 1$. Therefore $\exists \hat{k} \geq 0$ such that $\alpha^{(k)} < 1$ $(\forall k \geq \hat{k})$. Therefore by Lemma 2.1 for some norm $\|\cdot\|_* : M(R^n) \to R^n$, $\|M^{(k)}\|_* < 1$ $(\forall k \geq \hat{k})$, whence by (3.23)

$$\left\|\mathrm{w}\left(\underline{\mathbf{x}}^{(k+1)}\right)\right\|_* \leq \beta^{(k)}\left\|\mathrm{w}\left(\underline{\mathbf{x}}^{(k)}\right)\right\|_* \qquad (\forall k \geq \hat{k})$$

where $\beta^{(k)} = \left\|M^{(k)}\right\|_*^{m_k} \to 0$ $(k \to \infty)$. Therefore because all norms on R^n are topologically equivalent, it follows that $(\forall k \geq 0)$ $\exists c^{(k)} \geq 0$ such that $\left\|\mathrm{w}\left(\underline{\mathbf{x}}^{(k+1)}\right)\right\| \leq c^{(k)}\left\|\mathrm{w}\left(\underline{\mathbf{x}}^{(k)}\right)\right\|$ and $c^{(k)} \to 0$ $(k \to \infty)$. □

From (3.17)–(3.20), $\underline{\mathbf{v}}_n^{(k,m)} = \underline{\mathbf{u}}_n^{(k,m)}$, whence $\underline{\mathbf{x}}_n^{(k,m+1)} = \underline{\mathbf{y}}_n^{(k,m)}$. Furthermore $\underline{\mathbf{y}}_1^{(k,m+1)} \subseteq \underline{\mathbf{x}}_1^{(k,m+1)}$. This suggests the algorithm **INMSSS1**, which is obtained from **INSSS1** by setting $\underline{\mathbf{y}}_1^{(k,m)} = \underline{\mathbf{u}}_1^{(k,m)} = \underline{\mathbf{x}}_1^{(k,m)}$, $\underline{\mathbf{v}}_n^{(k,m)} = \underline{\mathbf{u}}_n^{(k,m)}$, and $\underline{\mathbf{x}}_n^{(k,m+1)} = \underline{\mathbf{y}}_n^{(k,m)}$ $(m = 0, \ldots, m_k - 1)$ $(\forall k \geq 0)$). It may be shown that theorems 3.7, 3.8 and 3.9 are valid for **INMSSS1**. The proofs are similar to those corresponding to **INSSS1**.

3.7 Numerical results

The algorithms which are described in this chapter, (together with others which computational experience shows to be inferior) have been implemented in Triplex S-algol [BaiC---a], [ColM-82a], [MorC-83a] on a Vax-11/780 computer. The convergence criteria used are $\|f(x^{(k)})\| \leq \varepsilon_1$ for the real algorithms and $\|\mathrm{w}(\underline{x}^{(k)})\| \leq \varepsilon_2$ for the interval algorithms. In the Newton algorithms the number of inner iterations m_k is given by $m_k = m_0 + [k/l]$ $(k \geq 0)$ where $[k/l]$ is the integer part of k/l .

To illustrate the effectiveness of the algorithms in this chapter results for Example 4 and Example 6, which are described in Appendix B, are given. For Example 4, if $S \subseteq R^n$ then $|d(x) - d(y)| \leq \mu |x - y|$ $(\forall x, y \in S)$ where

$$\mu = 6h^2 \max_{1 \leq i \leq n} \max_{x \in S} \left\{ (x_i - t_i/2 + 1)^2 \right\},$$

and if $\underline{\mathrm{d}} : I(R^n) \to I(R^n)$ is defined by

$$\underline{\mathrm{d}}_i(\underline{\mathrm{x}}) = 2h^2(\underline{\mathrm{x}}_i - t_i/2 + 1)^3 \qquad (i = 1, \ldots, n),$$

then $\mathrm{w}(\underline{\mathrm{d}}(\underline{\mathrm{x}})) \leq \nu \mathrm{w}(\underline{\mathrm{x}})$ $(\forall \underline{\mathrm{x}} \in I(S))$ where

$$\nu = 6h^2 \max_{1 \leq i \leq n} \max_{\underline{\mathrm{z}} \in I(S)} \left\{ |\underline{\mathrm{z}}_i - t_i/2 + 1|^2 \right\}.$$

Clearly both μ and ν are finite if $S \subset R^n$ is bounded and may be made arbitrarily small by making h sufficiently small. Furthermore if $A \in M(R^n)$ is as defined in Example 4 then by [OrtR-69a] 2.3.6, 2.3.14 A is an M-matrix whence by [OrtR-69a] 10.5.4 $\rho\left(D^{-1}B\right) < 1$. Similar considerations are valid for Example 6.

Tables 3.1–3.4 contain the CPU times, in seconds, required for the convergence criteria to be satisfied. In Table 3.1 the SSS2 and MSSS2 algorithms are compared with the algorithm RE [Ale--72a]. In Table 3.2 the NSSS1 and NMSSS1 algorithms are compared with the algorithm NRE*, which is obtained from the algorithm NRE [Ale--72a] by replacing m with m_k . In Table 3.3 the algorithms ISSS1 and IMSSS1 are compared with the algorithm REIDK [Ale--72a]. In Table 3.4 the algorithms INSSS1 and INMSSS1 are compared with the algorithm NREIDK* [Ale--72a].

The algorithms RE and REIDK require the solution of several nonlinear equations in one variable of the form $\alpha x + \beta d(x) = \gamma$. Several iterative algorithms were tested. The most efficient consists of computing $\tilde{x} = (\gamma - \beta d(\hat{x}))/\alpha$, where $\hat{x}$ is a given initial estimate of the required solution and $\tilde{x}$ is taken to be the final estimate of the solution. It can be shown that this method generates convergent sequences for RE and REIDK under easily satisfiable conditions.

The initial iterates are computed, as suggested by Alefeld [Ale--72a], from

$$x^{(0)} = A^{-1} \, (Td(0) + c)$$

for the real algorithms, and from

$$\underline{\mathbf{x}}^{(0)} = [-\alpha, \alpha]$$

where

$$\alpha = \left| A^{-1} \, (Td(0) + c) \right|,$$

for the interval algorithms.

The computational results which are contained in tables 3.1–3.4 together with results from several other numerical experiments suggest that the replacement of the single-step method with the symmetric single-step method can lead to significant reductions in CPU time.

Example	RE	SSS2	MSSS2
4 $\left(\begin{smallmatrix} n=10 \\ \varepsilon_1=10^{-6} \end{smallmatrix}\right)$	8.4	6.6	5.8
6 $\left(\begin{smallmatrix} n=16 \\ \varepsilon_1=10^{-6} \end{smallmatrix}\right)$	4.4	3.8	3.7

Table 3.1

Example	NRE*	NSSS1	NMSSS1
4 $\left(\begin{smallmatrix} n=10 \\ \varepsilon_1=10^{-6} \end{smallmatrix}\right)$	9.3	6.7	6.6
6 $\left(\begin{smallmatrix} n=16 \\ \varepsilon_1=10^{-6} \end{smallmatrix}\right)$	5.8	4.7	4.6

$m_0=10$ $l=5$

Table 3.2

Example	REIDK	ISSS1	IMSSS1
4 $\left(\begin{smallmatrix} n=10 \\ \varepsilon_2=10^{-2} \end{smallmatrix}\right)$	75.3	55.3	52.4
6 $\left(\begin{smallmatrix} n=16 \\ \varepsilon_2=10^{-4} \end{smallmatrix}\right)$	55.9	39.6	39.4

Table 3.3

Example	NREIDK*	INSSS1	INMSSS1
4 $\left(\begin{smallmatrix} n=10 \\ \varepsilon_2=10^{-2} \end{smallmatrix}\right)$	89.2	73.4	69.3
6 $\left(\begin{smallmatrix} n=16 \\ \varepsilon_2=10^{-4} \end{smallmatrix}\right)$	88.6	61.5	60.5

$m_0=5$ $l=20$

Table 3.4

4. The use of an inner iteration in the algorithm of Alefeld and Platzöder

The Krawczyk-Moore algorithm for bounding an isolated zero of a given mapping $f : R^n \to R^n$ using interval arithmetic [Kra--69a], [Moo--77a], [Moo--78a], has been shown to be very effective, especially when combined with a Moore and Jones search procedure [MooJ-77a], [Jon--78a], [Jon--80a]. Subsequently several authors have suggested modifications of the Krawczyk-Moore algorithm which improve computational efficiency.

Alefeld and Platzöder [AleP-83a] have introduced a quadratically convergent algorithm, similar to the Krawczyk-Moore algorithm, which as they point out, requires less computational labour per iteration than does the Krawczyk-Moore algorithm. In this thesis, KM and AP denote the Krawczyk-Moore and Alefeld-Platzöder algorithms respectively. A modification APSW, of the algorithm AP, which is similar to the modification KMW, of the algorithm KM, which has been suggested by Wolfe [Wol--80a], is described in this chapter.

4.1 Preliminaries

Let $A \in M(R^n)$ and $\underline{b} \in I(R^n)$ be given. Let $\underline{\mathcal{G}}(A, \underline{b}) \in I(R^n)$ be the result of applying the Gauss algorithm [AleH-83a] to the pair $(A, \underline{b})$. As explained by Alefeld and Platzöder [AleP-83a], there exist matrices $D_1, \ldots, D_n \in M(R^n)$, $T_1, \ldots, T_{n-1} \in M(R^n)$, $G_1, \ldots, G_{n-1} \in M(R^n)$ dependent only on A such that

$$\underline{\mathcal{G}}(A, \underline{b}) = D_1(T_1(\cdots T_{n-1}(D_n(G_{n-1}(\cdots(G_2(G_1\underline{b})\cdots). \tag{4.1}$$

Therefore

$$\begin{aligned} w\left(\underline{\mathcal{G}}(A, \underline{b})\right) &= |D_1||T_1|\cdots|T_{n-1}||D_n||G_{n-1}|\cdots|G_2||G_1|\, w(\underline{b}) \\ &= M_A\, w(\underline{b}) \end{aligned} \tag{4.2}$$

where $M_A \in M(R^n)$ depends only on A.

Let $\underline{K}_N : I(R^n) \times I(M(R^n)) \times M(R^n) \to I(R^n)$ be defined by

$$\underline{K}_N(\underline{x}, \underline{F}, A) = m(\underline{x}) - \underline{\mathcal{G}}(A, \{f(m(\underline{x})) - (A - \underline{F})(\underline{x} - m(\underline{x}))\}) \tag{4.3}$$

where $A \in M(R^n)$ is nonsingular, $\underline{\mathcal{G}} : M(R^n) \times I(R^n) \to I(R^n)$ is as described previously, and $f : R^n \to R^n$ is a given mapping. We have the following results.

Lemma 4.1 :

(a) Let $A \in M(R^n)$ be nonsingular. Then for $\underline{b} \in R^n$ arbitrary

$$A^{-1}\underline{b} \subseteq \underline{\mathcal{G}}(A, \underline{b}).$$

(b) Let $A \in M(R^n)$ be nonsingular and let $\underline{a}, \underline{b} \in I(R^n)$ be such that $w(\underline{a}) \leq \beta w(\underline{b})$ for some $\beta \geq 0$. Then

$$w\left(\underline{\mathcal{G}}(A, \underline{a})\right) \leq \beta w\left(\underline{\mathcal{G}}(A, \underline{b})\right).$$

Proof : The proof of Lemma 4.1 is identical to that of [AleP-83a] Lemma 1. □

Lemma 4.2 : Suppose that $f : D \subseteq R^n \to R^n$ is a given mapping with $f \in C^1(D)$. Let $\underline{f}' : I(D) \to I(M(R^n))$ be a continuous inclusion monotonic interval extension of $f' : D \to M(R^n)$. Let $\underline{x}^{(0)} \in I(D)$ be given with $w\left(\underline{x}^{(0)}\right) > 0$. If $B^{(0)} = m\left(\underline{f}'(\underline{x}^{(0)})\right)$ is nonsingular and

$$w\left(\underline{K}_N(\underline{x}^{(0)}, \underline{f}'(\underline{x}^{(0)}), B^{(0)})\right) \leq \alpha w\left(\underline{x}^{(0)}\right),$$

where $\alpha \in [0, 1)$, then $\underline{f}'(\underline{x}^{(0)})$ does not contain any singular matrices.

Proof : The proof of Lemma 4.2 is identical to that of [AleP-83a] Lemma 2. □

Lemma 4.3 :

(a) If $\underline{x}, \underline{y} \in I(R^n)$ and $\beta > 0$ are such that $w(\underline{y}) \leq \beta w(\underline{x})$, $\underline{F} \in I(M(R^n))$, $A \in M(R^n)$, and A^{-1} exists, then

$$w\left(\underline{K}_N(\underline{y}, \underline{F}, A)\right) \leq \beta w\left(\underline{K}_N(\underline{x}, \underline{F}, A)\right).$$

(b) If $\underline{\mathbf{F}}, \underline{\mathbf{G}} \in I(M(R^n))$ are such that $\underline{\mathbf{G}} \subseteq \underline{\mathbf{F}}$, $\underline{\mathbf{x}} \in I(R^n)$, $A \in M(R^n)$, and A^{-1} exists then

$$\mathrm{w}\left(\underline{\mathbf{K}}_N(\underline{\mathbf{x}}, \underline{\mathbf{G}}, A)\right) \leq \mathrm{w}\left(\underline{\mathbf{K}}_N(\underline{\mathbf{x}}, \underline{\mathbf{F}}, A)\right).$$

Proof :

(a) Let

$$\underline{\mathbf{a}} = f(\mathrm{m}(\underline{\mathbf{x}})) - (A - \underline{\mathbf{F}})(\underline{\mathbf{x}} - \mathrm{m}(\underline{\mathbf{x}})),$$

and

$$\underline{\mathbf{b}} = f(\mathrm{m}(\underline{\mathbf{x}})) - (A - \underline{\mathbf{F}})(\underline{\mathbf{y}} - \mathrm{m}(\underline{\mathbf{y}})).$$

If $\mathrm{w}(\underline{\mathbf{y}}) \leq \beta \mathrm{w}(\underline{\mathbf{x}})$, then $\mathrm{w}(\underline{\mathbf{b}}) \leq \beta \mathrm{w}(\underline{\mathbf{a}})$ and by Lemma 4.1(b),

$$\mathrm{w}\left(\underline{\mathcal{G}}(A, \underline{\mathbf{b}})\right) \leq \beta \mathrm{w}\left(\underline{\mathcal{G}}(A, \underline{\mathbf{a}})\right),$$

whence the result follows.

(b) Let

$$\underline{\mathbf{a}} = f(\mathrm{m}(\underline{\mathbf{x}})) - (A - \underline{\mathbf{F}})(\underline{\mathbf{x}} - \mathrm{m}(\underline{\mathbf{x}})),$$

and

$$\underline{\mathbf{b}} = f(\mathrm{m}(\underline{\mathbf{x}})) - (A - \underline{\mathbf{G}})(\underline{\mathbf{x}} - \mathrm{m}(\underline{\mathbf{x}})).$$

If $\underline{\mathbf{G}} \subseteq \underline{\mathbf{F}}$, then $\mathrm{w}(\underline{\mathbf{b}}) \leq \mathrm{w}(\underline{\mathbf{a}})$ and by Lemma 4.1(b) with $\beta = 1$,

$$\mathrm{w}\left(\underline{\mathcal{G}}(A, \underline{\mathbf{b}})\right) \leq \mathrm{w}\left(\underline{\mathcal{G}}(A, \underline{\mathbf{a}})\right),$$

whence the result follows. □

Lemma 4.4 : Suppose that $f : D \subseteq R^n \to R^n$ is a given mapping with $f \in C^1(D)$. Let $\underline{f}' : I(D) \to I(M(R^n))$ be an inclusion monotonic interval extension of $f' : D \to M(R^n)$. Suppose that $\exists \hat{D} \subseteq D$ such that $(\forall \underline{x} \in I(\hat{D}))$, $\mathrm{m}\left(\underline{f}'(\underline{x})\right)$ is nonsingular. Suppose also that $\exists \lambda > 0$ such that $(\forall \underline{x} \in I(\hat{D}))$,

$$\|\mathrm{w}\left(\underline{f}'(\underline{x})\right)\| \leq \lambda \|\mathrm{w}\left(\underline{x}\right)\|. \tag{4.4}$$

Then $\exists \mu > 0$ such that $(\forall \underline{x} \in I(D))$

$$\|\mathrm{w}\left(\underline{K}_N(\underline{x}, \underline{f}'(\underline{x}), \mathrm{m}\left(\underline{f}'(\underline{x})\right)\right)\| \leq \mu \|\mathrm{w}\left(\underline{x}\right)\|^2.$$

Proof : The proof of Lemma 4.4 is similar to the proof of part (b) of the theorem given by Alefeld and Platzöder [AleP-83a]. □

4.2 The algorithm APSW

Let $f : D \subseteq R^n \to R^n$ be a given mapping with $f \in C^1(\hat{D})$ where $\hat{D} \subset D$ is an open convex set. Let $\underline{f} : I(\hat{D}) \to I(R^n)$ and $\underline{f}' : I(\hat{D}) \to I(M(R^n))$ be continuous inclusion monotonic interval extensions of $f : \hat{D} \to R^n$ and $f' : \hat{D} \to M(R^n)$ respectively. Let $\hat{\underline{x}} \in I(D)$, a sequence of non-negative integers $p^{(k)}$, and $\alpha \in [0, 1)$ be given. Then the algorithm APSW for bounding an isolated zero of f in $\hat{\underline{x}}$ is as follows.

Algorithm 4.1 (APSW)

1. $\underline{x}^{(0)} := \hat{\underline{x}}$

2. $\underline{\mathbf{F}}^{(0)} := \underline{\mathbf{f}}'\left(\underline{\mathbf{x}}^{(0)}\right)$

3. $B^{(0)} := \mathrm{m}\left(\underline{\mathbf{F}}^{(0)}\right)$

4. $\underline{\mathbf{x}}^{(0,0)} := \underline{\mathbf{x}}^{(0)}$

5. for $m = 0$ to $p^{(0)}$ do

   5.1. $\underline{\mathbf{x}}^{(0,m+1)} := \underline{\mathbf{K}}_N\left(\underline{\mathbf{x}}^{(0,m)}, \underline{\mathbf{F}}^{(0)}, B^{(0)}\right) \cap \underline{\mathbf{x}}^{(0,m)}$

6. $\underline{\mathbf{x}}^{(1)} := \underline{\mathbf{x}}^{(0,p^{(0)}+1)}$

7. $k := 1$

8. while true do

   8.1. $\underline{\mathbf{F}}^{(k)} := \underline{\mathbf{f}}'\left(\underline{\mathbf{x}}^{(k)}\right)$

   8.2. $A^{(k)} := \mathrm{m}\left(\underline{\mathbf{F}}^{(k)}\right)$

   8.3. $\underline{\mathbf{u}}^{(k)} := \underline{\mathbf{K}}_N\left(\underline{\mathbf{x}}^{(k)}, \underline{\mathbf{F}}^{(k)}, A^{(k)}\right)$

   8.4. $\underline{\mathbf{z}}^{(k)} := \underline{\mathbf{u}}^{(k)} \cap \underline{\mathbf{x}}^{(k)}$

   8.5. if $\mathrm{w}\left(\underline{\mathbf{u}}^{(k)}\right) \leq \alpha \mathrm{w}\left(\underline{\mathbf{x}}^{(k)}\right)$

   then

   8.5.1. $B^{(k)} := A^{(k)}$

   8.5.2. $\underline{\mathbf{x}}^{(k,1)} := \underline{\mathbf{z}}^{(k)}$

   8.5.3. for $m = 1$ to $p^{(k)}$ do

   8.5.3.1. $\underline{\mathbf{u}}^{(k,m)} := \underline{\mathbf{K}}_N\left(\underline{\mathbf{x}}^{(k,m)}, \underline{\mathbf{F}}^{(k)}, A^{(k)}\right)$

   8.5.3.2. $\underline{\mathbf{x}}^{(k,m+1)} := \underline{\mathbf{u}}^{(k,m)} \cap \underline{\mathbf{x}}^{(k,m)}$

   8.5.4. $\underline{\mathbf{x}}^{(k+1)} := \underline{\mathbf{x}}^{(k,p^{(k)}+1)}$

   else

8.5.5. $B^{(k)} := B^{(k-1)}$

8.5.6. $\underline{\mathrm{Y}}^{(k)} := \underline{\mathbb{K}}_N\left(\underline{\mathrm{x}}^{(k)}, \underline{\mathrm{F}}^{(k)}, B^{(k)}\right)$

8.5.7. $\underline{\mathrm{x}}^{(k,1)} := \underline{\mathrm{Y}}^{(k)} \cap \underline{\mathrm{z}}^{(k)}$

8.5.8. for $m = 1$ to $p^{(k)}$ do

8.5.8.1. $\underline{\mathrm{Y}}^{(k,m)} := \underline{\mathbb{K}}_N\left(\underline{\mathrm{x}}^{(k,m)}, \underline{\mathrm{F}}^{(k)}, A^{(k)}\right)$

8.5.8.2. $\underline{\mathrm{x}}^{(k,m+1)} := \underline{\mathrm{Y}}^{(k,m)} \cap \underline{\mathrm{x}}^{(k,m)}$

8.5.9. $\underline{\mathrm{x}}^{(k+1)} := \underline{\mathrm{x}}^{(k,p^{(k)}+1)}$

8.6. $k := k + 1$ □

The algorithm APSW differs from the algorithm AP essentially in that the Jacobian is re-used $p^{(k)}$ times in the k^{th} iteration. If $p^{(k)} = 0 \quad (\forall k \geq 0)$ then APSW and AP are identical.

Theorem 4.1 : Let $f : D \subseteq R^n \to R^n$ be a given mapping with $f \in C^1(\hat{D})$ where $\hat{D} \subset D$ is an open convex set. Let $\underline{\mathrm{f}} : I(\hat{D}) \to I(R^n)$ and $\underline{\mathrm{f}}' : I(\hat{D}) \to I(M(R^n))$ be inclusion monotonic interval extensions of $f : \hat{D} \to R^n$ and $f' : \hat{D} \to M(R^n)$ respectively. Suppose that $\underline{\mathrm{x}}^{(0)} \in I(\hat{D})$ is such that $\mathrm{w}\left(\underline{\mathrm{x}}^{(0)}\right) > 0$, that $B^{(0)} = \mathrm{m}\left(\underline{\mathrm{f}}'(\underline{\mathrm{x}}^{(0)})\right)$ is nonsingular, and that for some $\alpha \in [0, 1)$

$$\mathrm{w}\left(\underline{\mathbb{K}}_N(\underline{\mathrm{x}}^{(0)}, \underline{\mathrm{f}}'(\underline{\mathrm{x}}^{(0)}), B^{(0)})\right) \leq \alpha \mathrm{w}\left(\underline{\mathrm{x}}^{(0)}\right). \tag{4.5}$$

If $\underline{\mathrm{x}}^{(0)}$ contains a zero x^* of f then the sequence $\left(\underline{\mathrm{x}}^{(k)}\right)$ which is generated from APSW is well defined and $\underline{\mathrm{x}}^{(k)} \to x^* \quad (k \to \infty)$. If also $\exists \lambda > 0$ such that

$(\forall \underline{x} \in I(\hat{D}))$

$$\|w(\underline{f}'(\underline{x}))\| \leq \lambda \|w(\underline{x})\|$$

then for each $p^{(k)} \geq 0$, $\exists \mu^{(k)} > 0$ such that

$$\left\|w\left(\underline{x}^{(k+1)}\right)\right\| \leq \mu^{(k)} \left\|w\left(\underline{x}^{(k)}\right)\right\|^{p^{(k)}+2} \qquad (\forall k \geq 0). \tag{4.6}$$

If APSW terminates because an empty intersection occurs in one of the steps 5.1, 8.4, 8.5.3.2, 8.5.7 or 8.5.8.2, then there is no zero of f in $\underline{x}^{(0)}$.

Finally if, instead of (4.5),

$$\underline{K}_N(\underline{x}^{(0)}, \underline{f}'(\underline{x}^{(0)}), B^{(0)}) \subset \text{int}(\underline{x}^{(0)}) \tag{4.7}$$

then (4.5) holds for some $\alpha \in [0,1)$, $\exists x^* \in \underline{x}^{(0)}$ such that $f(x^*) = 0$, and x^* is unique in $\underline{x}^{(0)}$.

Proof : From Lemma 4.2, $A^{(k)^{-1}}$ exists $(\forall k \geq 1)$, and therefore the sequence $\left(\underline{x}^{(k)}\right)$ generated from APSW is well defined provided no empty intersections occur.

The Krawczyk operator $\underline{K} : I(R^n) \times M(R^n) \to I(R^n)$ is defined by

$$\underline{K}(\underline{x}, Y) = m(\underline{x}) - Y f(m(\underline{x})) + (I - Y\underline{f}'(\underline{x}))(\underline{x} - m(\underline{x})).$$

By Lemma 4.1(a)

$$\underline{K}(\underline{x}^{(k)}, B^{(k)^{-1}}) \subseteq \underline{K}_N(\underline{x}^{(k)}, \underline{f}'(\underline{x}^{(k)}), B^{(k)})$$

so, if $x^* \in \underline{x}^{(k)}$, then $x^* \in \underline{x}^{(k,1)}$. By Lemma 4.1(a) and the inclusion monotonicity of $\underline{f}'$,

$$\underline{K}(\underline{x}^{(k,m)}, B^{(k)^{-1}}) \subseteq \underline{K}_N(\underline{x}^{(k,m)}, \underline{f}'(\underline{x}^{(k)}), B^{(k)})$$

whence $(x^* \in \underline{x}^{(k,m)}) \Rightarrow (x^* \in \underline{x}^{(k,m+1)})$. Therefore, by induction on m ,

$$(x^* \in \underline{x}^{(k)} \wedge f(x^*) = 0) \quad \Rightarrow \quad (x^* \in \underline{x}^{(k+1)}).$$

Now by induction on k ,

$$(x^* \in \underline{x}^{(0)} \wedge f(x^*) = 0) \quad \Rightarrow \quad (x^* \in \underline{x}^{(k)} \quad (\forall k \geq 0)).$$

From the way $\underline{x}^{(k,m)}$ is formed,

$$\underline{x}^{(k+1)} = \underline{x}^{(k,p^{(k)}+1)} \subseteq \underline{x}^{(k,p^{(k)})} \subseteq \cdots \subseteq \underline{x}^{(k,1)} \qquad (\forall k \geq 0). \tag{4.8}$$

We shall show that

$$\mathrm{w}\left(\underline{x}^{(k)}\right) \leq \alpha^k \, \mathrm{w}\left(\underline{x}^{(0)}\right) \qquad (\forall k \geq 1). \tag{4.9}$$

From (4.5), (4.9) holds for $k = 1$. Assume that (4.9) holds for all k such that $0 \leq k \leq \widehat{k}$. If $\underline{x}^{(\widehat{k}+1)}$ is generated from steps 8.5.5–8.5.9, then

$$\begin{aligned} \mathrm{w}\left(\underline{x}^{(\widehat{k},1)}\right) &\leq \mathrm{w}\left(\underline{\mathbf{K}}_N\left(\underline{x}^{(\widehat{k})}, \underline{\mathbf{f}}'(\underline{x}^{(\widehat{k})}), B^{(\widehat{k})}\right)\right) \\ &\leq \alpha \, \mathrm{w}\left(\underline{x}^{(\widehat{k})}\right) \\ &\leq \alpha^{\widehat{k}+1} \, \mathrm{w}\left(\underline{x}^{(0)}\right). \end{aligned}$$

So by (4.8),

$$\mathrm{w}\left(\underline{x}^{(\widehat{k}+1)}\right) \leq \alpha^{\widehat{k}+1} \, \mathrm{w}\left(\underline{x}^{(0)}\right). \tag{4.10}$$

If on the other hand, $\underline{x}^{(\widehat{k}+1)}$ is computed from steps 8.5.5–8.5.9, then from the way $B^{(k)}$ is formed, $\exists \widetilde{k}$ where $0 \leq \widetilde{k} \leq \widehat{k} - 1$ such that

$$B^{(\widehat{k})} = B^{(\widetilde{k})} = \mathrm{m}\left(\underline{\mathbf{f}}'(\underline{x}^{(\widetilde{k})})\right).$$

Now $\underline{\mathbf{x}}^{(\tilde{k}+1)}$ is computed from steps 8.5.1–8.5.4, so

$$\begin{aligned} \mathrm{w}\left(\underline{\mathbf{x}}^{(\tilde{k}+1)}\right) &\leq \mathrm{w}\left(\underline{\mathbf{K}}_N\left(\underline{\mathbf{x}}^{(\tilde{k})}, \underline{\mathbf{f}}'\left(\underline{\mathbf{x}}^{(\tilde{k})}\right), B^{(\tilde{k})}\right)\right) \\ &\leq \alpha\, \mathrm{w}\left(\underline{\mathbf{x}}^{(\tilde{k})}\right). \end{aligned} \tag{4.11}$$

We next show that

$$\mathrm{w}\left(\underline{\mathbf{x}}^{(\tilde{k}+j+1)}\right) \leq \alpha^{j+1}\, \mathrm{w}\left(\underline{\mathbf{x}}^{(\tilde{k})}\right) \qquad (j = 0, \ldots, \hat{k} - \tilde{k}). \tag{4.12}$$

By (4.11), (4.12) holds for $j = 0$. Now if (4.12) holds for some j, $(0 \leq j \leq \hat{k} - \tilde{k})$, then

$$\begin{aligned} \mathrm{w}\left(\underline{\mathbf{x}}^{(\tilde{k}+j+2)}\right) &\leq \mathrm{w}\left(\underline{\mathbf{K}}_N\left(\underline{\mathbf{x}}^{(\tilde{k}+j+1)}, \underline{\mathbf{f}}'\left(\underline{\mathbf{x}}^{(\tilde{k}+j+1)}\right), B^{(\tilde{k}+j+1)}\right)\right) \\ &\leq \alpha^{j+2}\, \mathrm{w}\left(\underline{\mathbf{x}}^{(\tilde{k})}\right). \end{aligned}$$

Therefore, by finite induction on j, (4.12) holds for $j = 0, \ldots, \hat{k} - \tilde{k}$. Therefore

$$\begin{aligned} \mathrm{w}\left(\underline{\mathbf{x}}^{(\hat{k}+1)}\right) &\leq \alpha^{\hat{k}-\tilde{k}+1}\, \mathrm{w}\left(\underline{\mathbf{x}}^{(\tilde{k})}\right) \\ &\leq \alpha^{\hat{k}+1}\, \mathrm{w}\left(\underline{\mathbf{x}}^{(0)}\right). \end{aligned} \tag{4.13}$$

Using (4.10) and (4.13), regardless of how $\underline{\mathbf{x}}^{(\hat{k}+1)}$ is computed, we have

$$\mathrm{w}\left(\underline{\mathbf{x}}^{(\hat{k}+1)}\right) \leq \alpha^{\hat{k}+1}\, \mathrm{w}\left(\underline{\mathbf{x}}^{(0)}\right)$$

and so (4.9) holds $(\forall k \geq 0)$. So $\mathrm{w}\left(\underline{\mathbf{x}}^{(k)}\right) \to 0 \quad (k \to \infty)$, whence, because $x^* \in \underline{\mathbf{x}}^{(k)} \quad (\forall k \geq 0)$, it follows that $\underline{\mathbf{x}}^{(k)} \to x^* \quad (k \to \infty)$.

Since

$$\underline{\mathbf{x}}^{(k,1)} \subseteq \underline{\mathbf{K}}_N\left(\underline{\mathbf{x}}^{(k)}, \underline{\mathbf{f}}'\left(\underline{\mathbf{x}}^{(k)}\right), A^{(k)}\right),$$

we have, by Lemma 4.4,

$$\left\| w\left(\underline{x}^{(k,1)}\right)\right\| \leq \mu \left\| w\left(\underline{x}^{(k)}\right)\right\|^2 .$$

Further, since $\underline{x}^{(k,m+1)} \subseteq \underline{K}_N(\underline{x}^{(k,m)}, \underline{f}'(\underline{x}^{(k)}), A^{(k)})$ it follows by a simple inductive argument that

$$\left\| w\left(\underline{x}^{(k+1)}\right)\right\| \leq \mu^{(k)} \left\| w\left(\underline{x}^{(k)}\right)\right\|^{p^{(k)}+2},$$

where

$$\mu^{(k)} = \mu \left\{(\lambda/2)\, \|M_{A^{(k)}}\|\right\}^{p^{(k)}}.$$

That there is no zero of f in $\underline{x}^{(0)}$ if APSW terminates due to an empty intersection follows from

$$\left(x^* \in \underline{x}^{(0)} \wedge f(x^*) = 0\right) \quad \Rightarrow \quad \left(x^* \in \underline{x}^{(k)} \quad (\forall k \geq 0)\right).$$

Finally, if instead of (4.5), (4.7) holds then clearly (4.5) holds with

$$\alpha = \max_{1 \leq i \leq n} \left\{ \left(w\left(\underline{K}_N(\underline{x}^{(0)}, \underline{f}'(\underline{x}^{(0)}), B^{(0)})\right)\right)_i \Big/ \left(w\left(\underline{x}^{(0)}\right)\right)_i \right\}.$$

Also since

$$\underline{K}(\underline{x}^{(0)}, \underline{B}^{(0)^{-1}}) \subseteq \underline{K}_N(\underline{x}^{(0)}, \underline{f}'(\underline{x}^{(0)}), B^{(0)}) \subset \operatorname{int}(\underline{x}^{(0)}),$$

from Lemma 2.13, $\exists x^* \in \underline{x}^{(0)}$ such that $f(x) = 0$, and x^* is unique in $\underline{x}^{(0)}$. □

Theoretical results similar to those given by Wolfe [Wol--80a] for KMW are valid with $p = p^{(k)}$.

4.3 Methods for determining the number of inner iterations in KMW and APSW

Computational experience with both KMW and APSW has shown that the choice of the number of inner iterations, $p^{(k)}$, greatly affects the efficiency of both algorithms. Wolfe [Wol--80a] has considered choosing $p^{(k)} = p$, $(\forall k \geq 0)$, for some fixed integer p in the implementation of KMW, and has suggested that the optimal value for p might be estimated using techniques similar to those described by Brent [Bre--73a]. This is however difficult to implement in practice. Hansen and Greenberg [HanG-83a] suggest re-using the Jacobian matrix while the width of successive iterates is being sufficiently reduced. Unfortunately it is difficult to determine what constitutes a sufficient width reduction since this varies greatly from problem to problem.

In this section two methods which automatically select $p^{(k)}$, independently of the size and complexity of the system of equations, are presented. Assume that the iterate $\underline{x}^{(k,m)}$ $(k \geq 0, m \geq 1)$ has been computed, and that a reliable estimate of the relative efficiencies of computing a new outer iterate, $\underline{x}^{(k+1,1)}$ and that of computing a new inner iterate, $\underline{x}^{(k,m+1)}$ can be obtained. Then whichever iterate is expected to be more efficient should be computed and the decision process repeated. Efficiency indices ρ_I and ρ_O corresponding to the computation of $\underline{x}^{(k,m+1)}$ and $\underline{x}^{(k+1,1)}$ respectively are given by

$$\rho_I = -\ln\left(\left\|w\left(\underline{x}^{(k,m+1)}\right)\right\| \Big/ \left\|w\left(\underline{x}^{(k,m)}\right)\right\|\right) \Big/ T_I$$

and

$$\rho_O = -\ln\left(\left\|w\left(\underline{x}^{(k+1,1)}\right)\right\| \Big/ \left\|w\left(\underline{x}^{(k,m)}\right)\right\|\right) \Big/ T_O,$$

where T_I and T_O are the CPU times required to compute $\underline{x}^{(k,m+1)}$ and

$\underline{x}^{(k+1,1)}$ respectively from $\underline{x}^{(k,m)}$. Experience with both KMW and APSW has shown that the CPU time required to compute each outer iterate does not vary greatly with k . Therefore the CPU time for the next outer iterate could be estimated by the CPU time for the previous outer iterate. Similarly the CPU time for the next inner iterate could be estimated by the CPU time required to compute the last inner iterate.

Since we have not yet decided whether to compute $\underline{x}^{(k,m+1)}$ or $\underline{x}^{(k+1,1)}$ at the time of the decision process, we do not know the values of $\left\|w\left(\underline{x}^{(k,m+1)}\right)\right\|$ and $\left\|w\left(\underline{x}^{(k+1,1)}\right)\right\|$ so these quantities must be estimated. From the theory for APSW,

$$\begin{aligned} w\left(\underline{x}^{(k,m+1)}\right) &\leq w\left(\underline{K}_N\left(\underline{x}^{(k,m)}, \underline{f}'\left(\underline{x}^{(k)}\right), A^{(k)}\right)\right) \\ &= M_{B^{(k)}}\left|A^{(k)} - \underline{f}'\left(\underline{x}^{(k)}\right)\right| w\left(\underline{x}^{(k,m)}\right). \end{aligned}$$

Therefore

$$\left\|w\left(\underline{x}^{(k,m+1)}\right)\right\| \leq \left\|M_{B^{(k)}}\right\| \left\|A^{(k)} - \underline{f}'\left(\underline{x}^{(k)}\right)\right\| \left\|w\left(\underline{x}^{(k,m)}\right)\right\|.$$

Similarly, if $\exists \lambda > 0$ such that $\left\|w\left(\underline{f}'(\underline{x})\right)\right\| \leq \lambda \left\|w(\underline{x})\right\| \quad (\forall \underline{x} \in I(\underline{x}^{(0)}))$ then,

$$\begin{aligned} w\left(\underline{x}^{(k+1,1)}\right) &\leq w\left(\underline{K}_N\left(\underline{x}^{(k+1)}, \underline{f}'\left(\underline{x}^{(k+1)}\right), A^{(k+1)}\right)\right) \\ &= \frac{1}{2} M_{A^{(k+1)}} w\left(\underline{f}'\left(\underline{x}^{(k+1)}\right)\right) w\left(\underline{x}^{(k+1)}\right), \end{aligned}$$

where $\underline{x}^{(k+1)} = \underline{x}^{(k,m)}$, whence

$$\begin{aligned} \left\|w\left(\underline{x}^{(k+1,1)}\right)\right\| &\leq \frac{1}{2}\left\|M_{A^{(k+1)}}\right\| \left\|w\left(\underline{f}'\left(\underline{x}^{(k+1)}\right)\right)\right\| \left\|w\left(\underline{x}^{(k+1)}\right)\right\| \\ &\leq \frac{\lambda}{2}\left\|M_{A^{(k+1)}}\right\| \left\|w\left(\underline{x}^{(k+1)}\right)\right\|^2. \end{aligned}$$

Similar results hold for KMW. This leads to the conjecture that $\|w(\underline{x}^{(k,m+1)})\|$ and $\|w(\underline{x}^{(k+1,1)})\|$ could be approximated by

$$\left\|w\left(\underline{x}^{(k,m+1)}\right)\right\| \approx N^{(k,m)}\left\|w\left(\underline{x}^{(k,m)}\right)\right\|$$

and

$$\left\|w\left(\underline{x}^{(k+1,1)}\right)\right\| \approx M^{(k)}\left\|w\left(\underline{x}^{(k,m)}\right)\right\|^2,$$

where

$$N^{(k,m)} = \left\|w\left(\underline{x}^{(k,m)}\right)\right\| \Big/ \left\|w\left(\underline{x}^{(k,m-1)}\right)\right\|$$

and

$$M^{(k)} = \left\|w\left(\underline{x}^{(k,1)}\right)\right\| \Big/ \left\|w\left(\underline{x}^{(k)}\right)\right\|^2.$$

The two strategies for deciding whether or not to compute another inner iterate use some or all of the above approximations and are as follows.

Strategy A

1. Compute the efficiency index ρ_I for an inner iteration from

$$\rho_I = -\ln\left(\left\|w\left(\underline{x}^{(k,m)}\right)\right\| \Big/ \left\|w\left(\underline{x}^{(k,m-1)}\right)\right\|\right) \Big/ T^{(k,m)}$$

where $T^{(k,m)}$ is the time required to compute $\underline{x}^{(k,m)}$ from $\underline{x}^{(k,m-1)}$.

2. Estimate the efficiency index ρ_O which would have been obtained if instead $\underline{x}^{(k+1,1)}$ had been computed from $\underline{x}^{(k+1)} = \underline{x}^{(k,m-1)}$ using

$$\rho_O \approx -\ln\left(\left(M^{(k)}\left\|w\left(\underline{x}^{(k,m-1)}\right)\right\|^2\right) \Big/ \left\|w\left(\underline{x}^{(k,m-1)}\right)\right\|\right) \Big/ T^{(k,1)}.$$

3. If $\rho_O > \rho_I$ then recompute the Jacobian. Otherwise re-use the Jacobian. □

Strategy B

1. Estimate the efficiency index ρ_I for the inner iterate from

$$\rho_I \approx -\ln\left(\left(N^{(k,m)}\left\|\mathrm{w}\left(\underline{x}^{(k,m)}\right)\right\|\right) / \left\|\mathrm{w}\left(\underline{x}^{(k,m)}\right)\right\|\right) \Big/ T^{(k,m)}.$$

2. Estimate the efficiency index ρ_O for computing the outer iterate $\underline{x}^{(k+1,1)}$ from $\underline{x}^{(k+1)} = \underline{x}^{(k,m)}$ using

$$\rho_O \approx -\ln\left(\left(M^{(k)}\left\|\mathrm{w}\left(\underline{x}^{(k,m)}\right)\right\|^2\right) / \left\|\mathrm{w}\left(\underline{x}^{(k,m)}\right)\right\|\right) \Big/ T^{(k,1)}.$$

3. If $\rho_O > \rho_I$ then recompute the Jacobian. Otherwise re-use the Jacobian. □

Although Strategy A should always give one inner iteration too many, it uses fewer approximations than does Strategy B. In practice there is little difference between the results obtained from the two strategies when they are applied to APSW and KMW.

4.4 Numerical results

The algorithms KMW and APSW have been implemented in Triplex S-algol [BaiC---a], [ColM-82a], [MorC-83a], using strategies A and B to determine the $p^{(k)}$ $(k \geq 0)$, and have been used to solve several systems of equations on a Vax-11/780 computer. Numerical results for examples 1, 2, 4, 5, and 7, which are described in Appendix B, are given. Tables 4.1–4.5 contain the CPU times, in seconds, which are

required for the algorithm KMW to converge for each of these examples. Tables 4.6–4.10 contain the CPU times, in seconds, which are required for the algorithm APSW to converge.

The numerical results which are presented in Tables 4.1–4.10 are obtained using the initial interval $\underline{x}^{(0)} = \left(\underline{x}_i^{(0)}\right)_{n\times 1}$ where for $i = 1,\ldots,n$, $\underline{x}_i^{(0)}$ has the value $[0,5]$, $[50,100]$, $[0,4]$, $[0,10]$, and $[1,7]$ for examples 1, 2, 4, 5, and 7 respectively. For each example (4.5) is satisfied with

$$\alpha = \max_{1\le i\le n} \left\{ w\left(\left(\underline{K}_N(\underline{x}^{(0)}, \underline{f}'(\underline{x}^{(0)}), B^{(0)}) \right)_i \right) / w\left(\underline{x}_i^{(0)}\right) \right\}$$

and (4.7) is also satisfied. The convergence criterion used for each example is $\|w\left(\underline{x}^{(k)}\right)\| \le 10^{-10}$.

The results given in tables 4.1–4.10 illustrate quite clearly that the algorithm APSW is more efficient than the algorithm KMW and the increase in efficiency becomes greater as n increases. The results also show that although the two strategies to select the number of inner iterations do not always choose the optimal value for $p^{(k)}$, they do always produce a very significant increase in efficiency over the algorithm AP, and this improvement increases with n . Thus of the four algorithms KM, AP, KMW, and APSW, APSW, using either strategy to determine the number of inner iterations $p^{(k)}$, is the most efficient.

n	A	B	$p^{(k)}=0$	$p^{(k)}=1$	$p^{(k)}=2$	$p^{(k)}=3$	$p^{(k)}=4$
5	46.55	52.64	52.68	47.48	48.53	50.29	52.27
10	153.74	153.84	174.98	141.87	136.75	144.44	145.35
20	605.96	520.16	772.83	598.44	540.51	445.11	434.73

Table 4.1 *KMW — Example 1*

n	A	B	$p^{(k)}=0$	$p^{(k)}=1$	$p^{(k)}=2$	$p^{(k)}=3$	$p^{(k)}=4$
5	30.06	30.66	40.30	30.02	32.63	29.80	29.58
10	76.17	84.00	101.93	73.82	81.30	72.45	72.83
20	380.82	289.93	478.24	348.59	370.79	285.99	237.18

Table 4.2 *KMW — Example 2*

n	A	B	$p^{(k)}=0$	$p^{(k)}=1$	$p^{(k)}=2$	$p^{(k)}=3$	$p^{(k)}=4$
5	21.93	23.77	24.49	22.50	23.44	25.19	23.93
10	82.21	75.61	83.96	73.38	71.94	78.77	74.77
20	495.43	450.03	922.34	688.09	501.41	535.31	466.16

Table 4.3 *KMW — Example 4*

n	A	B	$p^{(k)}=0$	$p^{(k)}=1$	$p^{(k)}=2$	$p^{(k)}=3$	$p^{(k)}=4$
5	22.22	22.48	25.36	22.15	23.41	23.98	25.39
10	77.09	73.18	91.03	74.77	78.79	76.93	78.49
20	510.70	542.44	960.83	626.15	566.86	506.66	488.94

Table 4.4 *KMW — Example 5*

n	A	B	$p^{(k)}=0$	$p^{(k)}=1$	$p^{(k)}=2$	$p^{(k)}=3$	$p^{(k)}=4$
9	37.92	44.20	44.79	36.33	43.96	38.56	41.92
16	202.59	224.07	296.36	197.98	211.50	181.76	183.00
25	532.40	498.73	1048.32	669.48	702.19	517.96	407.83

Table 4.5 *KMW — Example 7*

n	A	B	$p^{(k)}=0$	$p^{(k)}=1$	$p^{(k)}=2$	$p^{(k)}=3$	$p^{(k)}=4$
5	48.92	49.35	54.13	48.87	50.72	50.42	54.21
10	123.58	125.41	135.28	118.00	120.71	129.18	130.25
20	391.47	383.09	439.00	387.50	378.80	385.29	390.74

Table 4.6 *APSW — Example 1*

n	A	B	$p^{(k)}=0$	$p^{(k)}=1$	$p^{(k)}=2$	$p^{(k)}=3$	$p^{(k)}=4$
5	30.82	34.20	39.21	30.60	33.72	29.58	30.09
10	75.01	84.49	98.19	74.63	84.75	74.00	72.44
20	240.38	252.06	311.07	228.94	268.73	215.00	211.39

Table 4.7 *APSW — Example 2*

n	A	B	$p^{(k)}=0$	$p^{(k)}=1$	$p^{(k)}=2$	$p^{(k)}=3$	$p^{(k)}=4$
5	21.72	22.09	23.89	21.43	21.65	24.88	22.78
10	78.40	74.19	84.09	74.29	72.05	79.70	75.67
20	326.00	336.55	408.51	343.42	305.88	342.54	309.03

Table 4.8 *APSW — Example 4*

n	A	B	$p^{(k)}=0$	$p^{(k)}=1$	$p^{(k)}=2$	$p^{(k)}=3$	$p^{(k)}=4$
5	21.78	22.10	25.05	22.16	22.79	22.95	24.84
10	76.17	74.63	90.68	76.61	78.03	78.56	79.73
20	333.72	344.45	459.76	350.47	358.12	359.13	340.37

Table 4.9 *APSW — Example 5*

n	A	B	$p^{(k)}=0$	$p^{(k)}=1$	$p^{(k)}=2$	$p^{(k)}=3$	$p^{(k)}=4$
9	35.17	37.98	41.92	35.86	41.42	37.69	41.10
16	122.91	126.68	160.11	127.34	138.57	122.38	133.19
25	320.14	335.36	428.28	317.32	350.63	321.89	331.65

Table 4.10 *APSW — Example 7*

5. The Symmetric operator

Hansen and Sengupta [HanS-81a] have pointed out that the Krawczyk-Moore algorithm (KM), [Kra--69a], [Moo--77a], [Moo--78a], may be improved if the Krawczyk operator is replaced by what Moore and Qi [MooQ-82a] call the Hansen operator. This gives rise to the algorithm KMH for which Moore and Qi [MooQ-82a] have described some powerful computable existence, uniqueness, and convergence results. It is easy to show that the uniqueness and convergence results which are valid for KMH are also valid for the algorithm KMHW which is obtained when the modification described by Wolfe [Wol--80a] is used with the algorithm KMH.

It is shown in this chapter that if, in KMW, the Krawczyk operator is replaced with the Symmetric operator which is defined in Section 5.1, then an algorithm KMSW is obtained which is supported by a computationally effective existence, uniqueness, and convergence theory which permit it to be used with a Moore-Jones search procedure.

5.1 The Krawczyk, Hansen, and Symmetric operators

Let $f : D \subseteq R^n \rightarrow R^n$ be a given mapping with $f \in C^1(\hat{D})$ where $\hat{D} \subset D$

is an open convex set. Let $\underline{f}' : I(\hat{D}) \to I(M(R^n))$ be an inclusion monotonic interval extension of $f' : \hat{D} \to M(R^n)$. Let $Y \in M(R^n)$ be nonsingular and let

$$\underline{R}(\underline{x}) = I - Y\underline{f}'(\underline{x}) \tag{5.1}$$

$$= \left(\underline{r}_{ij}(\underline{x})\right)_{n\times n},$$

$$g(x) = Y f(x), \tag{5.2}$$

$$\underline{K}(\underline{x}) = \mathrm{m}\,(\underline{x}) - g(\mathrm{m}\,(\underline{x})) + \underline{R}(\underline{x})(\underline{x} - \mathrm{m}\,(\underline{x})), \tag{5.3}$$

$$\underline{H}_i(\underline{x}) = \mathrm{m}\,(\underline{x}_i) - g_i(\mathrm{m}\,(\underline{x})) + \sum_{j=1}^{i-1} \underline{r}_{ij}(\underline{x})(\underline{H}'_j(\underline{x}) - \mathrm{m}\,(\underline{x}_j))$$

$$+ \sum_{j=i}^{n} \underline{r}_{ij}(\underline{x})(\underline{x}_j - \mathrm{m}\,(\underline{x}_j)), \tag{5.4}$$

$$\underline{H}'_i(\underline{x}) = \underline{H}_i(\underline{x}) \cap \underline{x}_i \qquad (i = 1, \ldots, n), \tag{5.5}$$

$$\underline{S}_i(\underline{x}) = \mathrm{m}\,(\underline{x}_i) - g_i(\mathrm{m}\,(\underline{x})) + \sum_{j=1}^{i} \underline{r}_{ij}(\underline{x})(\underline{H}'_j(\underline{x}) - \mathrm{m}\,(\underline{x}_j))$$

$$+ \sum_{j=i+1}^{n} \underline{r}_{ij}(\underline{x})(\underline{S}'_j(\underline{x}) - \mathrm{m}\,(\underline{x}_j)), \tag{5.6}$$

$$\underline{S}'_i(\underline{x}) = \underline{S}_i(\underline{x}) \cap \underline{H}'_i(\underline{x}) \qquad (i = n, \ldots, 1). \tag{5.7}$$

Moore and Qi [MooQ-82a] refer to $\underline{K} : I(\hat{D}) \to I(R^n)$ and $\underline{H} : I(\hat{D}) \to I(R^n)$ as the Krawczyk and Hansen operators respectively. Moore [Moo-77a,] [Moo--78a], and Qi [Qi---80a] have shown that, under appropriate hypotheses, if $x^* \in \hat{\underline{x}}$ and $f(x^*) = 0$ then $x^* \in \underline{K}(\hat{\underline{x}})$, that if $\underline{K}(\hat{\underline{x}}) \subseteq \hat{\underline{x}}$ then $\exists x^* \in \hat{\underline{x}}$ such that $f(x^*) = 0$, and that if the sequence $\left(y^{(k)}\right)$ is generated from

$$y^{(k+1)} = y^{(k)} - Y f(y^{(k)}) \qquad (\forall k \geq 0) \tag{5.8}$$

with $y^{(0)} = \mathrm{m}\,(\hat{\underline{x}})$, then $y^{(k)} \to x^* \quad (k \to \infty)$, and also that if $\underline{K}(\hat{\underline{x}}) \subset \mathrm{int}(\hat{\underline{x}})$

then x^* is unique in $\hat{\underline{x}}$ and if $\left(y^{(k)}\right)$ is generated from (5.8) with $y^{(0)} \in \hat{\underline{x}}$ arbitrary then $y^{(k)} \to x^* \quad (k \to \infty)$.

Moore and Qi [MooQ-82a] have also shown that if $\underline{H}'(\hat{\underline{x}}) \neq \emptyset$, $x^* \in \hat{\underline{x}}$, and $f(x^*) = 0$, then $x^* \in \underline{H}'(\hat{\underline{x}})$, and that if $\underline{H}(\hat{\underline{x}}) \neq \emptyset$ and $\underline{H}(\hat{\underline{x}}) \subseteq \hat{\underline{x}}$ then $\exists x^* \in \underline{H}'(\hat{\underline{x}})$ such that $f(x^*) = 0$, and if $\left(y^{(k)}\right)$ is generated from (5.8) with $y^{(0)} = \mathrm{m}(\hat{\underline{x}})$, then $y^{(k)} \to x^* \quad (k \to \infty)$. Moore and Qi [MooQ-82a] have shown furthermore that if $\underline{H}(\hat{\underline{x}}) \neq \emptyset$, $\underline{H}(\hat{\underline{x}}) \subseteq \hat{\underline{x}}$, and $\mathrm{w}(\underline{H}(\hat{\underline{x}})) < \mathrm{w}(\hat{\underline{x}})$, then $\exists x^* \in \underline{H}(\hat{\underline{x}})$ such that $f(x^*) = 0$, x^* is unique in $\hat{\underline{x}}$, and if $\left(y^{(k)}\right)$ is generated from (5.8) with $y^{(0)} \in \underline{H}(\hat{\underline{x}})$ arbitrary, then $y^{(k)} \to x^* \quad (k \to \infty)$. Additional results of this sort have been given by Qi [Qi---80a].

5.2 Preliminaries

The following results are used subsequently.

Lemma 5.1 : Let $\underline{x} \in I(\hat{D})$ be given. Then

(a) $\underline{H}(\underline{x}) \subseteq \underline{K}(\underline{x})$;

(b) $\underline{S}(\underline{x}) \subseteq \underline{H}(\underline{x})$;

(c) $(\underline{S}(\underline{x}) \subseteq \underline{x}) \quad \Rightarrow \quad (\underline{S}(\underline{x}) \subseteq \underline{H}'(\underline{x}))$;

(d) $((\underline{S}(\underline{x}) \subset \mathrm{int}(\underline{x}) \quad \wedge \quad (\exists j_i \in \{i, \ldots, n\}, \quad |\underline{r}_{ij_i}(\underline{x})| \neq 0 \quad (i = 1, \ldots, n)) \quad \wedge$
$(Y = \mathrm{m}(\underline{f}'(\underline{x})))) \quad \Rightarrow \quad (\mathrm{w}(\underline{S}(\underline{x})) < \mathrm{w}(\underline{H}'(\underline{x})))$.

Proof :

(a) This result follows immediately from (5.3), (5.4) and (5.5).

(b) This result follows immediately from (5.4), (5.5), (5.6) and (5.7).

(c) This result follows immediately from Lemma 5.1(b).

(d) If $\underline{\mathbb{S}}(\underline{x}) \subset \mathrm{int}(\underline{x})$ then

$$|\underline{\mathbb{S}}(\underline{x}) - \mathrm{m}(\underline{x})| < |\underline{x} - \mathrm{m}(\underline{x})|. \tag{5.9}$$

Furthermore, since $Y = (\mathrm{m}(\underline{\mathbb{f}}'(\underline{x})))^{-1}$, it follows that $\underline{\mathbb{R}}(\underline{x})$ is symmetric, and $\underline{\mathbb{S}}'(\underline{x}) = \underline{\mathbb{S}}(\underline{x})$. Therefore for $i, j = 1, \ldots, n$, if $\mathrm{w}(\underline{r}_{ij}(\underline{x})) \neq 0$, then

$$\begin{aligned} \mathrm{w}(\underline{r}_{ij}(\underline{x})(\underline{\mathbb{S}}'_j(\underline{x}) - \mathrm{m}(\underline{x}_j))) &= \mathrm{w}(\underline{r}_{ij}(\underline{x}))\,|\underline{\mathbb{S}}'_j(\underline{x}) - \mathrm{m}(\underline{x}_j)| \\ &< \mathrm{w}(\underline{r}_{ij}(\underline{x}))\,|\underline{x}_j - \mathrm{m}(\underline{x}_j)|. \end{aligned} \tag{5.10}$$

Now if $\exists j_i \in \{i+1, \ldots, n\}$ such that $|\underline{r}_{ij_i}(\underline{x})| \neq 0$, then by (5.10),

$$\begin{aligned} \mathrm{w}(\underline{\mathbb{H}}_i(\underline{x})) &= \sum_{j=1}^{i-1} \mathrm{w}(\underline{r}_{ij}(\underline{x}))\,|\underline{\mathbb{H}}'_j(\underline{x}) - \mathrm{m}(\underline{x}_j)| + \sum_{j=i}^{n} \mathrm{w}(\underline{r}_{ij}(\underline{x}))\,|\underline{x}_j - \mathrm{m}(\underline{x}_j)| \\ &> \sum_{j=1}^{i} \mathrm{w}(\underline{r}_{ij}(\underline{x}))\,|\underline{\mathbb{H}}'_j(\underline{x}) - \mathrm{m}(\underline{x}_j)| + \sum_{j=i+1}^{n} \mathrm{w}(\underline{r}_{ij}(\underline{x}))\,|\underline{\mathbb{S}}'_j(\underline{x}) - \mathrm{m}(\underline{x}_j)| \\ &= \mathrm{w}(\underline{\mathbb{S}}_i(\underline{x})). \end{aligned}$$

Therefore $\mathrm{w}(\underline{\mathbb{S}}_i(\underline{x})) < \mathrm{w}(\underline{\mathbb{H}}'_i(\underline{x}))$ since $\underline{\mathbb{S}}_i(\underline{x}) \subset \mathrm{int}(\underline{x}_i)$. Suppose, on the other hand, that $|\underline{r}_{ij}(\underline{x})| = 0 \quad (j = i+1, \ldots, n)$ and that $|\underline{r}_{ii}(\underline{x})| \neq 0$. If $\underline{\mathbb{H}}'_i(\underline{x}) \not\subset \mathrm{int}(\underline{x}_i)$ then $\mathrm{w}(\underline{\mathbb{S}}_i(\underline{x})) < \mathrm{w}(\underline{\mathbb{H}}'_i(\underline{x}))$ since $\underline{\mathbb{S}}_i(\underline{x}) \subseteq \underline{\mathbb{H}}'_i(\underline{x})$ and $\underline{\mathbb{S}}_i(\underline{x}) \subset \mathrm{int}(\underline{x}_i)$. Conversely, if $\underline{\mathbb{H}}'_i(\underline{x}) \subset \mathrm{int}(\underline{x}_i)$ then $|\underline{\mathbb{H}}'_i(\underline{x}) - \mathrm{m}(\underline{x}_i)| < |\underline{x}_i - \mathrm{m}(\underline{x}_i)|$, whence

$$\begin{aligned} \mathrm{w}(\underline{\mathbb{H}}'_i(\underline{x})) &= \mathrm{w}(\underline{\mathbb{H}}_i(\underline{x})) \\ &= \sum_{j=1}^{i-1} \mathrm{w}(\underline{r}_{ij}(\underline{x}))\,|\underline{\mathbb{H}}'_j(\underline{x}) - \mathrm{m}(\underline{x}_j)| + \mathrm{w}(\underline{r}_{ii}(\underline{x}))\,|\underline{x}_i - \mathrm{m}(\underline{x}_i)| \\ &> \sum_{j=1}^{i} \mathrm{w}(\underline{r}_{ij}(\underline{x}))\,|\underline{\mathbb{H}}'_j(\underline{x}) - \mathrm{m}(\underline{x}_j)| \\ &= \mathrm{w}(\underline{\mathbb{S}}_i(\underline{x})). \end{aligned}$$

as required. □

Lemma 5.2 : Let $P : D \to R^n$ be defined by

$$P(x) = x - Yf(x). \tag{5.11}$$

Then

(a) $(x \in \underline{\mathbb{S}}(\underline{\mathbf{x}}) \quad \wedge \quad \underline{\mathbb{S}}(\underline{\mathbf{x}}) \subseteq \underline{\mathbb{H}}'(\underline{\mathbf{x}})) \quad \Rightarrow \quad (P(x) \in \underline{\mathbb{S}}(\underline{\mathbf{x}}))$;

(b) $(x \in \underline{\mathbf{x}}) \quad \Rightarrow \quad (P(x) \in \underline{\mathbb{K}}(\underline{\mathbf{x}}))$;

(c) $(x^* \in \underline{\mathbf{x}} \quad \wedge \quad f(x^*) = 0) \quad \Rightarrow \quad (x^* \in \underline{\mathbb{S}}'(\underline{\mathbf{x}}))$.

Proof :

(a) Let

$$e_{ij} = \begin{cases} 1 & (i = j) \\ 0 & (i \neq j) \end{cases} \qquad (i, j = 1, \ldots, n).$$

Then from Lemma 2.9, $\exists \theta_k \in [0, 1] \quad (k = 1, \ldots, n)$ such that if $\xi_k = x_k + \theta_k(x_k - \mathrm{m}(\underline{\mathbf{x}}_k)) \quad (k = 1, \ldots, n)$ then, for $j = 1, \ldots, n$,

$$f_j(x) - f_j(\mathrm{m}(\underline{\mathbf{x}})) = \sum_{k=1}^{n} \partial_k f_j(\xi_k)(x_k - \mathrm{m}(\underline{\mathbf{x}}_k)).$$

Therefore for $i = 1, \ldots, n$,

$$\begin{aligned} P_i(x) &= x_i - g_i(x) \\ &= \mathrm{m}(\underline{\mathbf{x}}_i) - g_i(\mathrm{m}(\underline{\mathbf{x}})) + (x_i - \mathrm{m}(\underline{\mathbf{x}}_i)) - \sum_{j=1}^{n} Y_{ij}(f_j(x) - f_j(\mathrm{m}(\underline{\mathbf{x}}))) \\ &= \mathrm{m}(\underline{\mathbf{x}}_i) - g_i(\mathrm{m}(\underline{\mathbf{x}})) + \sum_{k=1}^{n} e_{ik}(x_k - \mathrm{m}(\underline{\mathbf{x}}_k)) \end{aligned}$$

$$- \sum_{j=1}^{n} Y_{ij} \sum_{k=1}^{n} \partial_k f_j(\xi_k)(x_k - \mathrm{m}(\underline{\mathrm{x}}_k))$$

$$= \mathrm{m}(\underline{\mathrm{x}}_i) - g_i(\mathrm{m}(\underline{\mathrm{x}})) + \sum_{k=1}^{n} (e_{ik} - \sum_{j=1}^{n} Y_{ij} \partial_k f_j(\xi_k))(x_k - \mathrm{m}(\underline{\mathrm{x}}_k)). \quad (5.12)$$

Therefore since $\xi_k \in \underline{\mathrm{x}}_k \quad (k = 1, \ldots, n)$, and $x_k \in \underline{\mathbb{S}}_k(\underline{\mathrm{x}}) \quad (k = 1, \ldots, n)$, it follows that

$$P_i(x) \in \mathrm{m}(\underline{\mathrm{x}}_i) - g_i(\mathrm{m}(\underline{\mathrm{x}})) + \sum_{k=1}^{n} (I - Y\underline{\mathrm{f}}'(\underline{\mathrm{x}}))_{ik} (\underline{\mathbb{S}}_k(\underline{\mathrm{x}}) - \mathrm{m}(\underline{\mathrm{x}}_k))$$

$$\subseteq \mathrm{m}(\underline{\mathrm{x}}_i) - g_i(\mathrm{m}(\underline{\mathrm{x}})) + \sum_{k=1}^{i} \underline{\mathrm{r}}_{ik}(\underline{\mathrm{x}})(\underline{\mathrm{H}}'_k(\underline{\mathrm{x}}) - \mathrm{m}(\underline{\mathrm{x}}_k))$$

$$+ \sum_{k=i+1}^{n} \underline{\mathrm{r}}_{ik}(\underline{\mathrm{x}})(\underline{\mathbb{S}}'_k(\underline{\mathrm{x}}) - \mathrm{m}(\underline{\mathrm{x}}_k))$$

$$= \underline{\mathbb{S}}_i(\underline{\mathrm{x}}) \quad (i = 1, \ldots, n).$$

(b) From (5.12), for $i = 1, \ldots, n$,

$$P_i(x) = \mathrm{m}(\underline{\mathrm{x}}_i) - g_i(\mathrm{m}(\underline{\mathrm{x}})) + \sum_{k=1}^{n} (e_{ik} - \sum_{j=1}^{n} Y_{ij} \partial_k f_j(\xi_k))(x_k - \mathrm{m}(\underline{\mathrm{x}}_k))$$

$$\in \mathrm{m}(\underline{\mathrm{x}}_i) - g_i(\mathrm{m}(\underline{\mathrm{x}})) + \sum_{k=1}^{n} \underline{\mathrm{r}}_{ik}(\underline{\mathrm{x}})(\underline{\mathrm{x}}_k - \mathrm{m}(\underline{\mathrm{x}}_k)), \quad (5.13)$$

whence $P_i(x) \in \underline{\mathbb{K}}_i(\underline{\mathrm{x}})$, $(i = 1, \ldots, n)$.

(c) From (5.11) and (5.13),

$$x_1^* = P_1(x^*)$$

$$\in \mathrm{m}(\underline{\mathrm{x}}_1) - g_1(\mathrm{m}(\underline{\mathrm{x}})) + \sum_{k=1}^{n} \underline{\mathrm{r}}_{1k}(\underline{\mathrm{x}})(\underline{\mathrm{x}}_k - \mathrm{m}(\underline{\mathrm{x}}_k))$$

$$= \underline{\mathrm{H}}_1(\underline{\mathrm{x}}),$$

whence $x_1^* \in \underline{H}_1'(\underline{x})$. Suppose that for some $i \geq 2$, $x_j^* \in \underline{H}_j'(\underline{x})$ $(j = 1, \ldots, i-1)$. Then by (5.12),

$$\begin{aligned} x_i^* &= \mathrm{m}(\underline{x}_i) - g_i(\mathrm{m}(\underline{x})) + \sum_{k=1}^{n}(e_{ik} - \sum_{j=1}^{n} Y_{ij}\partial_k f_j(\xi_k))(x_k^* - \mathrm{m}(\underline{x}_k)) \\ &\in \mathrm{m}(\underline{x}_i) - g_i(\mathrm{m}(\underline{x})) + \sum_{k=1}^{i-1} \underline{r}_{ik}(\underline{x})(\underline{H}_k'(\underline{x}) - \mathrm{m}(\underline{x}_k)) + \sum_{k=i}^{n} \underline{r}_{ik}(\underline{x})(\underline{x}_k - \mathrm{m}(\underline{x}_k)) \\ &= \underline{H}_i(\underline{x}), \end{aligned}$$

whence $x_i^* \in \underline{H}_i'(\underline{x})$, and so by finite induction on i , $x^* \in \underline{H}'(\underline{x})$. A similar inductive argument now shows that $x^* \in \underline{S}'(\underline{x})$. □

Lemma 5.3 : Let $\hat{\underline{x}} \in I(\hat{D})$ be given and let $\underline{K}^* : I(\hat{D}) \to I(M(R^n))$ be defined by

$$\underline{K}^*(\underline{x}) = \mathrm{m}(\underline{x}) - Yf(\mathrm{m}(\underline{x})) + \underline{R}(\hat{\underline{x}})(\underline{x} - \mathrm{m}(\underline{x})). \tag{5.14}$$

If $\underline{x} \subseteq \hat{\underline{x}}$ then $\underline{K}(\underline{x}) \subseteq \underline{K}^*(\underline{x})$.

Proof : This result follows from (5.3), (5.14) and the inclusion monotonicity of $\underline{f}'$. □

5.3 Existence

The following theorem contains a sufficient condition for the existence of a zero of a given mapping f in a given box $\hat{\underline{x}}$ which is weaker than the condition $\underline{H}(\hat{\underline{x}}) \subseteq \hat{\underline{x}}$ of Moore and Qi [MooQ-82a].

Theorem 5.1 : Let $f : D \subseteq R^n \to R^n$ be a given mapping with $f \in C^1(\hat{D})$ where $\hat{D} \subset D$ is an open convex set, and let $\hat{\underline{x}} \in I(\hat{D})$ be given. If (i) $Y = \{\mathrm{m}(\underline{f}'(\hat{\underline{x}}))\}^{-1}$ exists; (ii) $\underline{S}(\hat{\underline{x}}) \neq \emptyset$; (iii) $\underline{S}(\hat{\underline{x}}) \subseteq \hat{\underline{x}}$, then $\exists x^* \in \underline{S}(\hat{\underline{x}})$ such that $f(x^*) = 0$.

Proof : By Lemma 5.1(c), $\underline{S}(\hat{\underline{x}}) \in \underline{H}'(\hat{\underline{x}})$, so by Lemma 5.2(a), if $x \in \underline{S}(\hat{\underline{x}})$ then $P(x) \in \underline{S}(\hat{\underline{x}})$. Therefore by Lemma 2.12, $\exists x^* \in \underline{S}(\hat{\underline{x}})$ such that $f(x) = 0$. □

By Lemma 5.1(b), $\underline{S}(\hat{\underline{x}}) \subseteq \underline{H}(\hat{\underline{x}})$ whether or not $\underline{H}(\hat{\underline{x}}) \subseteq \hat{\underline{x}}$, so it is possible that $\underline{S}(\hat{\underline{x}}) \subseteq \hat{\underline{x}}$ and $\underline{H}(\hat{\underline{x}}) \not\subseteq \hat{\underline{x}}$. Therefore Theorem 5.1 contains an existence test which is weaker than the test $\underline{H}(\hat{\underline{x}}) \subseteq \hat{\underline{x}}$. The additional computational labour which is required to evaluate $\underline{S}(\hat{\underline{x}})$ after evaluating $\underline{H}'(\hat{\underline{x}})$ is considerably reduced if use is made of the fact that the right-hand sides of (5.4) and (5.6) contain a sum over $i - 1$ terms in common.

5.4 Uniqueness

The following theorems contain sufficient conditions for the uniqueness of a zero of a given mapping f in a given initial box $\hat{\underline{x}}$ which are likely to be weaker, in most cases, than the conditions $\underline{H}(\hat{\underline{x}}) \subseteq \hat{\underline{x}}$ and $\mathrm{w}(\underline{H}(\hat{\underline{x}})) < \mathrm{w}(\hat{\underline{x}})$ of Moore and Qi [MooQ-82a].

Theorem 5.2 : Let $f : D \subseteq R^n \to R^n$ be a given mapping with $f \in C^1(\hat{D})$ where $\hat{D} \subset D$ is an open convex set and let $\hat{\underline{x}} \in I(\hat{D})$ be given. If (i) $Y =$

$\{\mathrm{m}(\underline{f}'(\hat{\underline{x}}))\}^{-1}$ exists; (ii) $\underline{S}(\hat{\underline{x}}) \neq \emptyset$; (iii) $\underline{S}(\hat{\underline{x}}) \subseteq \hat{\underline{x}}$; (iv) $\|\underline{R}(\hat{\underline{x}})\| < 1$, then $\exists x^* \in \underline{S}(\hat{\underline{x}})$ such that $f(x^*) = 0$ and x^* is unique in $\hat{\underline{x}}$.

Proof : Let the sequence $(\underline{x}^{(k)})$ be generated from

$$\underline{x}^{(0)} = \underline{S}(\hat{\underline{x}}), \tag{5.15}$$

$$\underline{x}^{(k+1)} = \underline{S}(\hat{\underline{x}}) \cap \underline{K}^*(\underline{x}^{(k)}) \qquad (\forall k \geq 0). \tag{5.16}$$

By Theorem 5.1, $\exists x^* \in \underline{x}^{(0)}$ such that $f(x^*) = 0$. Suppose that for some $k \geq 0$, $x^* \in \underline{x}^{(k)}$. By Lemma 5.2(b), $x^* = P(x^*) \in \underline{K}(\underline{x}^{(k)})$, so by Lemma 5.3, $x^* \in \underline{K}^*(\underline{x}^{(k)})$, whence $x^* \in \underline{x}^{(k+1)}$. Therefore by induction on k , $x^* \in \underline{x}^{(k)}$ $(\forall k \geq 0)$. Further, for $i = 1, \ldots, n$,

$$\begin{aligned} \mathrm{w}\left(\underline{x}_i^{(k+1)}\right) &\leq \mathrm{w}\left(\underline{K}_i^*(\underline{x}^{(k)})\right) \\ &\leq \|\underline{R}(\hat{\underline{x}})\| \left\|\mathrm{w}\left(\underline{x}^{(k)}\right)\right\| \qquad (\forall k \geq 0). \end{aligned}$$

Therefore

$$\begin{aligned} \left\|\mathrm{w}\left(\underline{x}^{(k+1)}\right)\right\| &\leq \|\underline{R}(\hat{\underline{x}})\| \left\|\mathrm{w}\left(\underline{x}^{(k)}\right)\right\| \\ &\leq \|\underline{R}(\hat{\underline{x}})\|^{k+1} \left\|\mathrm{w}\left(\underline{x}^{(0)}\right)\right\| \qquad (\forall k \geq 0), \end{aligned}$$

so if $\|\underline{R}(\hat{\underline{x}})\| < 1$, then $\mathrm{w}\left(\underline{x}^{(k)}\right) \to 0$ $(k \to \infty)$. Now $x^* \in \underline{x}^{(k)}$ $(\forall k \geq 0)$, so x^* is unique in $\underline{S}(\hat{\underline{x}})$ and therefore by Lemma 5.2(c) x^* is unique in $\hat{\underline{x}}$. □

Theorem 5.3 : Let $f : D \subseteq R^n \to R^n$ be a given mapping with $f \in C^1(\hat{D})$ where $\hat{D} \subset D$ is an open convex set. If (i) $Y = \{\mathrm{m}(\underline{f}'(\hat{\underline{x}}))\}^{-1}$ exists; (ii) $\underline{S}(\hat{\underline{x}}) \neq \emptyset$; (iii) $\underline{S}(\hat{\underline{x}}) \subseteq \hat{\underline{x}}$; (iv) $\mathrm{w}(\underline{S}(\hat{\underline{x}})) < \mathrm{w}(\underline{H}'(\underline{x}))$, then $\exists x^* \in \underline{S}(\hat{\underline{x}})$ such that $f(x^*) = 0$ and x^* is unique in $\hat{\underline{x}}$.

Proof : Let $\left(\underline{\mathrm{x}}^{(k)}\right)$ be generated from (5.15) and (5.16). Then as in the proof of Theorem 5.2, $\exists x^* \in \underline{\mathrm{x}}^{(0)}$ such that $f(x^*) = 0$ and $x^* \in \underline{\mathrm{x}}^{(k)}$ $(\forall k \geq 0)$. By (5.16), for $i = 1, \ldots, n$,

$$\mathrm{w}\left(\underline{\mathrm{x}}_i^{(k+1)}\right) \leq \sum_{j=1}^{n} \mathrm{w}\left(\underline{\mathrm{r}}_{ij}(\hat{\underline{\mathrm{x}}})\right) \mathrm{w}\left(\underline{\mathrm{x}}_j^{(k)}\right) / 2. \tag{5.17}$$

Furthermore for $i = 1, \ldots, n$,

$$\begin{aligned} \mathrm{w}\left(\underline{\mathrm{S}}_i(\hat{\underline{\mathrm{x}}})\right) &= \sum_{j=1}^{i} \mathrm{w}\left(\underline{\mathrm{r}}_{ij}(\hat{\underline{\mathrm{x}}})\right) \left|\underline{\mathrm{H}}'_j(\hat{\underline{\mathrm{x}}}) - \mathrm{m}\left(\hat{\underline{\mathrm{x}}}_j\right)\right| + \sum_{j=i+1}^{n} \mathrm{w}\left(\underline{\mathrm{r}}_{ij}(\hat{\underline{\mathrm{x}}})\right) \left|\underline{\mathrm{S}}'_j(\hat{\underline{\mathrm{x}}}) - \mathrm{m}\left(\hat{\underline{\mathrm{x}}}_j\right)\right| \\ &\geq \sum_{j=1}^{n} \mathrm{w}\left(\underline{\mathrm{r}}_{ij}(\hat{\underline{\mathrm{x}}})\right) \mathrm{w}\left(\underline{\mathrm{S}}_j(\hat{\underline{\mathrm{x}}})\right) / 2. \end{aligned} \tag{5.18}$$

Let

$$\beta = \max_{1 \leq i \leq n} \left\{ \mathrm{w}\left(\underline{\mathrm{S}}_i(\hat{\underline{\mathrm{x}}})\right) / \mathrm{w}\left(\underline{\mathrm{H}}'_i(\hat{\underline{\mathrm{x}}})\right) \right\}.$$

Then $\beta \in [0, 1)$. It will be shown that $(\forall k \geq 0)$,

$$\mathrm{w}\left(\underline{\mathrm{x}}^{(kn)}\right) \leq \beta^k \mathrm{w}\left(\underline{\mathrm{S}}(\hat{\underline{\mathrm{x}}})\right). \tag{5.19}$$

By (5.15), (5.19) holds for $k = 0$. Suppose that (5.19) holds for some $k \geq 0$. It may be shown by finite induction on l that for $l = 0, \ldots, n$,

$$\mathrm{w}\left(\underline{\mathrm{x}}_i^{(kn+l)}\right) \leq \beta^k \mathrm{w}\left(\underline{\mathrm{S}}_i(\hat{\underline{\mathrm{x}}})\right) \qquad (i = 1, \ldots, n - l) \tag{5.20}$$

and

$$\mathrm{w}\left(\underline{\mathrm{x}}_i^{(kn+l)}\right) \leq \beta^{k+1} \mathrm{w}\left(\underline{\mathrm{S}}_i(\hat{\underline{\mathrm{x}}})\right) \quad (i = n - l + 1, \ldots, n). \tag{5.21}$$

The argument is similar to that used by Moore and Qi [MooQ-82a]. That (5.19) holds for $k + 1$ follows from (5.21) with $l = n$. Therefore by induction on k , (5.19) holds $(\forall k \geq 0)$ whence $\mathrm{w}\left(\underline{\mathrm{x}}^{(k)}\right) \to 0$ $(k \to \infty)$. But $x^* \in \underline{\mathrm{x}}^{(k)}$ $(\forall k \geq 0)$, so x^* is unique in $\hat{\underline{\mathrm{x}}}$ by Lemma 5.2(c). □

Corollary 5.1 : If the hypotheses of Theorem 5.3 are valid save that instead of (iii) and (iv), $\underline{\mathbb{S}}(\hat{\underline{x}}) \subset \text{int}(\hat{\underline{x}})$ and for $i = 1, \ldots, n$, $\exists j_i \in \{i, \ldots, n\}$ such that $|\underline{r}_{ij_i}(\hat{\underline{x}})| \neq 0$, then $\exists x^* \in \underline{\mathbb{S}}(\hat{\underline{x}})$ such that $f(x^*) = 0$, and x^* is unique in $\hat{\underline{x}}$.

Proof : By Lemma 5.1(d), $\text{w}(\underline{\mathbb{S}}(\hat{\underline{x}})) < \text{w}(\underline{H}'(\hat{\underline{x}}))$. The result which it is required to prove now follows from Theorem 5.3. □

The uniqueness test which is contained in Corollary 5.1 is of course more stringent than the one contained in Theorem 5.3. Furthermore, if $\underline{r}_{ij}(\hat{\underline{x}}) = [r_{ijI}, r_{ijS}]$ then what is computed is actually $[\bar{r}_{ijI}, \bar{r}_{ijS}]$ where $\bar{r}_{ijI}$ and $\bar{r}_{ijS}$ are machine numbers and $\bar{r}_{ijI} \leq r_{ijI} \leq r_{ijS} \leq \bar{r}_{ijS}$. Therefore it is impossible to computationally verify that $|\underline{r}_{ij}(\hat{\underline{x}})| \neq 0$.

5.5 Convergence

In this section convergence results similar to those which are due to Moore and Qi [MooQ-82a] are given.

Theorem 5.4 : If the hypotheses of Theorem 5.1 are valid and if $Y = \{\text{m}(\underline{f}'(\hat{\underline{x}}))\}^{-1}$ exists then $\exists x^* \in \hat{\underline{x}}$ such that $f(x^*) = 0$, and the sequence $(y^{(k)})$ generated from (5.8) with $y^{(0)} = \text{m}(\hat{\underline{x}})$ converges to x^* .

Proof : Let $\underline{\mathbb{S}}^* \in I(R^n)$ be defined by

$$\text{m}(\underline{\mathbb{S}}^*) = \text{m}(\hat{\underline{x}})$$

and

$$w(\underline{\mathbb{S}}^*) = w(\underline{\mathbb{S}}(\hat{\underline{x}})) + 2|g(m(\hat{\underline{x}}))|$$

where $g : R^n \to R^n$ is given by (5.2). It follows, from a similar argument to that which is given by Moore and Qi [MooQ-82a] that $\underline{\mathbb{S}}(\hat{\underline{x}}) \subseteq \underline{\mathbb{S}}^* \subseteq \hat{\underline{x}}$ and $\underline{\mathbb{K}}(\underline{\mathbb{S}}^*) \subseteq \underline{\mathbb{S}}^*$, whence the result follows from Lemma 2.13. □

Theorem 5.5 : If the hypotheses of Theorem 5.2 are valid and $(y^{(k)})$ is generated from (5.8) with $y^{(0)} \in \underline{\mathbb{S}}(\hat{\underline{x}})$ arbitrary, then $y^{(k)} \to x^*$ $(k \to \infty)$ where x^* is the unique zero of f in $\hat{\underline{x}}$.

Proof : Let the sequence $(\underline{x}^{(k)})$ be generated from (5.15) and (5.16). It follows as in the proof of Theorem 5.2, that $\exists x^* \in \underline{\mathbb{S}}(\hat{\underline{x}})$ such that $f(x^*) = 0$, that $x^* \in \underline{x}^{(k)}$ $(\forall k \geq 0)$, and that $\underline{x}^{(k)} \to x^*$ $(k \to \infty)$. By (5.15) $y^{(0)} \in \underline{x}^{(0)}$. Suppose that $y^{(k)} \in \underline{x}^{(k)}$ for some $k \geq 0$. Then by lemmas 5.1(c) and 5.2(a), $y^{(k+1)} = P(y^{(k)}) \in \underline{\mathbb{S}}(\hat{\underline{x}})$ and by Lemma 5.2(b) and Lemma 5.3, $y^{(k+1)} = P(y^{(k)}) \in \underline{\mathbb{K}}^*(\underline{x}^{(k)})$. Therefore, by (5.16), $y^{(k+1)} \in \underline{x}^{(k+1)}$ whence by induction on k, $y^{(k)} \in \underline{x}^{(k)}$ $(\forall k \geq 0)$. Therefore $y^{(k)} \to x^*$ $(k \to \infty)$. □

Theorem 5.6 : If the hypotheses of Theorem 5.3 are valid and $(y^{(k)})$ is generated from (5.8) with $y^{(0)} \in \underline{\mathbb{S}}(\hat{\underline{x}})$ arbitrary, then $y^{(k)} \to x^*$ $(k \to \infty)$, where x^* is the unique zero of f in $\hat{\underline{x}}$.

Proof : As in the proof of Theorem 5.5, if $(\underline{x}^{(k)})$ is generated from (5.15), then

$y^{(k)} \in \underline{x}^{(k)} \quad (\forall k \geq 0)$, whence by Theorem 5.3, $y^{(k)} \to x^* \quad (k \to \infty)$. □

Corollary 5.2 : If the hypotheses of Corollary 5.1 are valid then $\exists x^* \in \underline{S}(\hat{\underline{x}})$ such that $f(x^*) = 0$, x^* is the unique zero of f in $\hat{\underline{x}}$, and if $\left(y^{(k)}\right)$ is generated from (5.8) with $y^{(0)} \in \underline{S}(\hat{\underline{x}})$ arbitrary, then $y^{(k)} \to x^* \quad (k \to \infty)$.

Proof : The hypotheses of Theorem 5.3 are valid, so the result follows from Theorem 5.6. □

5.6 The algorithm KMSW

It is shown in this section that if, in Wolfe's modification, KMW of the Krawczyk-Moore algorithm, the Krawczyk operator is replaced with the Symmetric operator, then an algorithm KMSW is obtained to which some of the results which have been developed in sections 5.1–5.4 are applicable. This permits KMSW to be used in a Moore-Jones search procedure.

The algorithm KMSW consists of generating the sequence $\left(\underline{x}^{(k)}\right)$ with $\underline{x}^{(0)} \in I(R^n)$ given as follows.

Algorithm 5.1 (KMSW)

1. $\underline{F}^{(0)} := \underline{f}'(\underline{x}^{(0)})$

2. $B^{(0)} := \left\{ \mathrm{m}\left(\underline{\mathbf{F}}^{(0)} \right) \right\}^{-1}$

3. $A^{(0)} := B^{(0)}$

4. $\underline{\mathbf{R}}^{(0)} := I - A^{(0)} \underline{\mathbf{F}}^{(0)}$

5. $r^{(0)} := \left\| \underline{\mathbf{R}}^{(0)} \right\|$

6. $k := 0$

7. while true do

   7.1. $\underline{\mathbf{x}}^{(k,0)} := \underline{\mathbf{x}}^{(k)}$

   7.2. for $m = 0$ to $p^{(k)}$ do

      7.2.1. $x^{(k,m)} := \mathrm{m}\left(\underline{\mathbf{x}}^{(k,m)} \right)$

      7.2.2. $g^{(k,m)} := A^{(k)} f\left(x^{(k,m)} \right)$

      7.2.3. for $i = 1$ to n do

         7.2.3.1. $$\underline{\mathbf{H}}_i^{(k,m)} := x_i^{(k,m)} - g_i^{(k,m)} + \sum_{j=1}^{i-1} \underline{\mathbf{R}}_{ij}^{(k)} \left(\underline{\mathbf{H}}'^{(k,m)}_j - x_j^{(k,m)} \right) + \sum_{j=i}^{n} \underline{\mathbf{R}}_{ij}^{(k)} \left(\underline{\mathbf{x}}_j^{(k,m)} - x_j^{(k,m)} \right)$$

         7.2.3.2. $\underline{\mathbf{H}}'^{(k,m)}_i := \underline{\mathbf{H}}_i^{(k,m)} \cap \underline{\mathbf{x}}_i^{(k,m)}$

      7.2.4. for $i = n$ to 1 by -1 do

         7.2.4.1. $$\underline{\mathbf{S}}_i^{(k,m)} := x_i^{(k,m)} - g_i^{(k,m)} + \sum_{j=1}^{i} \underline{\mathbf{R}}_{ij}^{(k)} \left(\underline{\mathbf{H}}'^{(k,m)}_j - x_j^{(k,m)} \right) + \sum_{j=i+1}^{n} \underline{\mathbf{R}}_{ij}^{(k)} \left(\underline{\mathbf{S}}'^{(k,m)}_j - x_j^{(k,m)} \right)$$

         7.2.4.2. $\underline{\mathbf{S}}'^{(k,m)}_i := \underline{\mathbf{S}}_i^{(k,m)} \cap \underline{\mathbf{H}}'^{(k,m)}_i$

      7.2.5. $\underline{\mathbf{x}}^{(k,m+1)} := \underline{\mathbf{S}}'^{(k,m)}$

7.3. $\underline{x}^{(k+1)} := \underline{x}^{(k,p^{(k)}+1)}$

7.4. $\underline{\mathbb{F}}^{(k+1)} := \underline{f}'\left(\underline{x}^{(k+1)}\right)$

7.5. $B^{(k+1)} := \left\{\mathrm{m}\left(\underline{\mathbb{F}}^{(k+1)}\right)\right\}^{-1}$

7.6. $\underline{\mathbb{R}}^{(k+1)} := I - B^{(k+1)}\underline{\mathbb{F}}^{(k+1)}$

7.7. $r^{(k+1)} := \left\|\underline{\mathbb{R}}^{(k+1)}\right\|$

7.8. if $r^{(k+1)} \leq r^{(k)}$

then

7.8.1. $A^{(k+1)} := B^{(k+1)}$

else

7.8.2. $A^{(k+1)} := A^{(k)}$

7.8.3. $\underline{\mathbb{R}}^{(k+1)} := I - A^{(k+1)}\underline{\mathbb{F}}^{(k+1)}$

7.8.4. $r^{(k+1)} := \left\|\underline{\mathbb{R}}^{(k+1)}\right\|$

7.9. $k := k + 1$ □

The preceding pseudo-code form of KMSW should, of course, be modified in obvious ways to obtain a usable implementation. For example, steps 7.2.3.1 and 7.2.4.1 contain a sum of terms corresponding to $j = 1, \ldots, i-1$ in common and this should be computed once only. Furthermore following step 7.2.5, $\underline{x}^{(k,m+1)}$ should be tested for convergence, using, for example, the criterion $\left\|\mathrm{w}\left(\underline{x}^{(k,m+1)}\right)\right\| \leq \varepsilon$ where $\varepsilon > 0$ is provided by the user. Some provision must also be made for determining $p^{(k)}$. It is preferable to use one of the strategies for determining $p^{(k)}$ automatically in each iteration which have been described in Chapter 4.

The computational value of KMSW is greatly enhanced by the following theorem, which permits KMSW to be used with a Moore-Jones search whereby a given box in $I(R^n)$ can be systematically divided into sub-boxes, each of which can be tested for the existence of a unique zero of f to which the KMSW sequence converges.

Theorem 5.7 : Let $f : D \subseteq R^n \to R^n$ be a given mapping, with $f \in C^1(\hat{D})$, where $\hat{D} \subseteq D$ is an open convex set. Let $\underline{f} : I(\hat{D}) \to I(M(R^n))$ be a continuous inclusion monotonic interval extension of the derivative $f' : \hat{D} \to M(R^n)$ of f. Let $\underline{x}^{(0)} \in I(\hat{D})$ be given. If (i) $B^{(0)}$ exists; (ii) $\underline{S}^{(0,0)} \subseteq \underline{x}^{(0)}$ and $\underline{S}^{(0,0)} \neq \emptyset$; (iii) $w\left(\underline{S}^{(0,0)}\right) < w\left(\underline{H}'^{(0,0)}\right)$, then $\exists x^* \in \underline{S}^{(0,0)}$ such that $f(x^*) = 0$, x^* is unique in $\underline{x}^{(0)}$, and if $\left(\underline{x}^{(k)}\right)$ is generated from KMSW then $x^* \in \underline{x}^{(k+1)} \subseteq \underline{x}^{(k)}$ $(\forall k \geq 0)$. Furthermore if (iv) $r^{(0)} < 1$, then $\underline{x}^{(k)} \to x^*$ $(k \to \infty)$ and

$$\left\| w\left(\underline{x}^{(k+1)}\right) \right\| \leq \left(r^{(0)}\right)^{p^{(k)}+1} \left\| w\left(\underline{x}^{(k)}\right) \right\| \qquad (\forall k \geq 0) \tag{5.22}$$

where $p^{(k)} \geq 0$ $(\forall k \geq 0)$.

Proof : The existence and uniqueness results follow from Theorem 5.3. It remains to show that $x^* \in \underline{x}^{(k+1)} \subseteq \underline{x}^{(k)}$ $(\forall k \geq 0)$, that $\underline{x}^{(k)} \to x^*$ $(k \to \infty)$, and that (5.22) holds.

Suppose that for some $k \geq 0$ and some $m \geq 0$, $x^* \in \underline{x}^{(k,m)}$. Then by Lemma 5.2(c), $x^* \in \underline{S}'^{(k,m)}$ whence $x^* \in \underline{x}^{(k,m+1)}$. Therefore by finite induction on m, $\left(x^* \in \underline{x}^{(k)} = \underline{x}^{(k,0)}\right) \Rightarrow \left(x^* \in \underline{x}^{(k+1)}\right)$, whence by induction on k,

$x^* \in \underline{x}^{(k)} \quad (\forall k \geq 0)$, and so, by inspection of KMSW, $x^* \in \underline{x}^{(k+1)} \subseteq \underline{x}^{(k)}$ $(\forall k \geq 0)$.

Now by steps 7.2.3.1 and 7.2.3.2 of KMSW, for $m = 0, \ldots, p^{(k)} \quad (\forall k \geq 0)$,

$$\mathrm{w}\left(\underline{\mathrm{H}}^{\prime(k,m)}\right) \leq \left|\underline{\mathrm{R}}^{(k)}\right| \mathrm{w}\left(\underline{\mathrm{x}}^{(k,m)}\right),$$

and by step 7.2.4.1 of KMSW, for $m = 0, \ldots, p^{(k)} \quad (\forall k \geq 0)$

$$\mathrm{w}\left(\underline{\mathrm{S}}^{(k,m)}\right) \leq \left|\underline{\mathrm{R}}^{(k)}\right| \mathrm{w}\left(\underline{\mathrm{x}}^{(k,m)}\right),$$

whence for $m = 0, \ldots, p^{(k)} \quad (\forall k \geq 0)$

$$\mathrm{w}\left(\underline{\mathrm{x}}^{(k,m)}\right) \leq \left|\underline{\mathrm{R}}^{(k)}\right|^m \mathrm{w}\left(\underline{\mathrm{x}}^{(k,0)}\right).$$

Therefore $(\forall k \geq 0)$

$$\mathrm{w}\left(\underline{\mathrm{x}}^{(k+1)}\right) \leq \left|\underline{\mathrm{R}}^{(k)}\right|^{p^{(k)}+1} \mathrm{w}\left(\underline{\mathrm{x}}^{(k)}\right),$$

whence $(\forall k \geq 0)$

$$\left\|\mathrm{w}\left(\underline{\mathrm{x}}^{(k+1)}\right)\right\| \leq \left\|\underline{\mathrm{R}}^{(k)}\right\|^{p^{(k)}+1} \left\|\mathrm{w}\left(\underline{\mathrm{x}}^{(k)}\right)\right\|. \tag{5.23}$$

Now by step 7.8 of KMSW, $\left\|\underline{\mathrm{R}}^{(k+1)}\right\| \leq \left\|\underline{\mathrm{R}}^{(k)}\right\| \quad (\forall k \geq 0)$, so (5.22) holds. Therefore if $r^{(0)} < 1$ then $\underline{\mathrm{x}}^{(k)} \to x^* \quad (\forall k \geq 0)$. □

The following theorem shows that under very easily satisfied hypotheses, the sequence $\left(\underline{\mathrm{x}}^{(k)}\right)$ generated from KMSW can converge very rapidly.

Theorem 5.8 : If hypotheses (i)–(iv) of Theorem 5.7 are valid and if also (v) $\exists \lambda > 0$ such that $\|w(\underline{f}'(\underline{x}))\| \leq \lambda \|w(\underline{x})\| \quad (\forall \underline{x} \in I(\widehat{D}))$; (vi) $\exists \mu > 0$ such that $A^{(k)} = \{m(\underline{f}'(\underline{x}^{(k)}))\}^{-1} + E^{(k)}$, where $\|E^{(k)}\| \leq \mu \|w(\underline{x}^{(k)})\| \quad (\forall k \geq 0)$, then $(\forall k \geq 0)$, $\exists \alpha^{(k)} > 0$ such that

$$\left\|w\left(\underline{x}^{(k+1)}\right)\right\| \leq \alpha^{(k)} \left\|w\left(\underline{x}^{(k)}\right)\right\|^{p^{(k)}+2}.$$

Proof : By the definition of $r^{(k)}$ and hypotheses (v) and (vi), $(\forall k \geq 0)$

$$\begin{aligned} r^{(k)} &= \left\|I - A^{(k)}\underline{f}'(\underline{x}^{(k)})\right\| \\ &\leq \left\|E^{(k)}\underline{f}'(\underline{x}^{(k)})\right\| + \left\|\left\{m\left(\underline{f}'(\underline{x}^{(k)})\right)\right\}^{-1}\right\| \left\|w\left(\underline{f}'(\underline{x}^{(k)})\right)\right\| \quad / \quad 2 \\ &\leq \left\{\mu \left\|\underline{f}'(\underline{x}^{(0)})\right\| + \lambda\nu/2\right\} \left\|w\left(\underline{x}^{(k)}\right)\right\|, \end{aligned} \tag{5.24}$$

where

$$\nu = \sup\left\{\|A^{-1}\| \quad | \quad A \in \underline{f}'(\underline{x}^{(0)})\right\}.$$

Let

$$\alpha^{(k)} = \left\{\mu \left\|\underline{f}'(\underline{x}^{(0)})\right\| + \lambda\nu/2\right\}^{p^{(k)}+1} (k \geq 0).$$

Then by (5.23),(5.24), $(\forall k \geq 0)$

$$\left\|w\left(\underline{x}^{(k+1)}\right)\right\| \leq \alpha^{(k)} \left\|w\left(\underline{x}^{(k)}\right)\right\|^{p^{(k)}+2}. \quad \square$$

5.7 Numerical results

The existence, uniqueness, and convergence tests described in this chapter have been incorporated into a search procedure of the type described by Moore and Jones

[MooJ-77a], and by Jones [Jon--78a], [Jon--80a]. Given a mapping $f : R^n \to R^n$ and a box $\underline{x} \in I(R^n)$ the search procedure determines the sub-boxes of $\underline{x}$ which are *safe*. A box $\hat{\underline{x}} \in I(R^n)$ is considered *safe* if and only if it satisfies at least one of the following tests.

1. $\underline{H}(\hat{\underline{x}}) \subseteq \hat{\underline{x}} \quad \wedge \quad \|\underline{R}(\hat{\underline{x}})\| < 1,$
2. $\underline{H}(\hat{\underline{x}}) \subset \text{int}(\hat{\underline{x}}),$
3. $\underline{H}(\hat{\underline{x}}) \subseteq \hat{\underline{x}} \quad \wedge \quad w(\underline{H}(\hat{\underline{x}})) < w(\hat{\underline{x}})$
4. $\underline{S}(\hat{\underline{x}}) \subseteq \hat{\underline{x}} \quad \wedge \quad \|\underline{R}(\hat{\underline{x}})\| < 1,$
5. $\underline{S}(\hat{\underline{x}}) \subseteq \hat{\underline{x}} \quad \wedge \quad w(\underline{S}(\hat{\underline{x}})) < w(\underline{H}'(\hat{\underline{x}}))$.

If a box $\hat{\underline{x}}$ is not *safe* then either it is rejected because it contains no zero of f or it is bisectd along a given co-ordinate direction using the maximum width rule described by Jones [Jon--78a] to give $\underline{x}' \in I(R^n)$ and $\underline{x}'' \in I(R^n)$. Both $\underline{x}'$ and $\underline{x}''$ are then examined in order to determine whether or not they are *safe*. Which of $\underline{x}'$ and $\underline{x}''$ is examined first is determined by the random region selection rule of Jones [Jon--78a]. In using tests 1, 2, and 3 however, if the test is not satisfied but $m(\hat{\underline{x}}) \notin \underline{H}'(\hat{\underline{x}})$, the box $\hat{\underline{x}}$ is not bisected and $\underline{H}'(\hat{\underline{x}})$ is taken as the next box to be examined. Similarly, in using tests 4 and 5, $\hat{\underline{x}}$ is not bisected and $\underline{S}'(\hat{\underline{x}})$ is the next box to be examined if $m(\hat{\underline{x}}) \notin \underline{S}'(\hat{\underline{x}})$. This procedure is found to be more effective than bisection.

The search procedure has been implemented in Triplex S-algol [BaiC---a], [ColM-82a], [MorC-83a] on a Vax-11/780 computer. Any of the five tests could be selected. Numerical results for Example 7 and Example 8, which are described in Appendix B, are given in this section. The results for Example 7 are obtained with $n = 9$ and the initial box is a 9-cube centred at the origin with radius 10. The results

for Example 8 are obtained with $n = 3$ and the initial box is a 3-cube centred at the origin with radius 1.

Table 5.1 contains the CPU times, in seconds required for the search procedure when each of the tests 1–5 are used. Table 5.2 contains the number of sub-boxes examined during the search.

Example	Test 1	Test 2	Test 3	Test 4	Test 5
7	142.0	142.0	142.0	124.0	124.0
8	30.7	31.1	31.3	29.6	29.6

Table 5.1 *CPU Times*

Example	Test 1	Test 2	Test 3	Test 4	Test 5
7	13	13	13	11	11
8	37	37	37	34	34

Table 5.2 *Boxes Examined*

When, in Example 7, $\hat{\underline{x}} \in I(R^9)$ with $\hat{\underline{x}}_i = [-7.5, 10.5]$ $(i = 1, \ldots, n)$ is subjected to each of the tests 1–5 in turn, tests 1–3 fail since $\underline{H}(\hat{\underline{x}}) \not\subseteq \hat{\underline{x}}$, Test 4 fails

since $\|\underline{R}(\hat{\underline{x}})\| \geq 1$. Test 5 however accepts $\hat{\underline{x}}$ as a *safe* box. Thus it is possible for Test 5 to be satisfied while tests 1–4 are not.

The search procedure which is described in this section has been used to determine *safe* boxes for several mappings. Of the numerical experiments which have been performed none has indicated that any of the tests 1, 2, and 3 require less CPU time than tests 4 and 5, and the superiority of tests 4 and 5 increases with n . The results which are least favourable for tests 4 and 5 all correspond to exampes where $n = 2$; this is to be expected, as can be seen from (5.1)–(5.7).

Once tests 1,2 and 3 have been implemented, tests 4 and 5 can be implemented with little additional programming. Computational experience with several mappings indicates that, for $n > 2$, tests 4 and 5 are preferable to tests 1, 2 and 3.

If, for a given box $\hat{\underline{x}}$, Test 4 is satisfied, then Theorem 5.7 guarantees that the algorithm KMSW with $\underline{x}^{(0)} = \hat{\underline{x}}$ will converge to the unique zero x^* of f in $\hat{\underline{x}}$. In order to illustrate the effectiveness of the algorithm KMSW, the algorithms KMW, KMHWand KMSW, the algorithm HS of Hansen and Sengupta [HanS-81a], and the algorithm HG of Hansen and Greenberg [HanG-83a], have been implemented in Triplex S-algol [BaiC---a], [ColM-82a], [MorC-83a] on a Vax-11/780 computer. The number of inner iterations $p^{(k)}$ for KMW, KMHW and KMSW can either be chosen using Strategy B, described in Chapter 4, or can be assigned a fixed value p , provided by the user. If in KMW and KMHW, $p^{(k)} = 0$ $(\forall k \geq 0)$ then the algorithms KM and KMH respectively are obtained.

Tables 5.3–5.7 contain the CPU times, in seconds, which are required to attain

numerical convergence for examples 1, 2, 4, 5 and 7, described in Appendix B, respectively. The convergence criterion is $\|w(\underline{x}^{(k)})\| \leq 10^{-10}$. The initial boxes are given by $\underline{x}^{(0)} = \left(\underline{x}_i^{(0)}\right)_{n\times 1}$, where for $i = 1, \ldots, n$ $\underline{x}_i^{(0)}$ has the value $[0, 5]$, $[50, 100]$, $[0, 4]$, $[0, 10]$ and $[1, 7]$ for examples 1, 2, 4, 5 and 7 respectively.

Table 5.8 contains the results which are obtained by using the algorithm HG on Example 2. In Table 5.8, s is the significant box width improvement factor [HanG-83a] which must be selected by the user. The program which implements HG has been used on several examples. In particular the results for the Broyden banded function (Appendix B—Example 8), presented in [HanG-83a], have been closely reproduced. Although the algorithm HG can be effective for small values of n , and with the correct choice for s , it does not in general appear to be competitive with the other algorithms. This is illustrated by Table 5.8 which should be compared with Table 5.4.

Of the algorithms KMW, KMHW, KMSW and HS, the algorithm KMSW using Strategy B for the automatic determination of the $p^{(k)}$ would appear to be the most efficient, especially for large n , where the saving in CPU time is most apparent.

n	KMW $p^{(k)}=0$	KMW B	KMHW $p^{(k)}=0$	KMHW B	KMSW $p^{(k)}=0$	KMSW B	HS $p^{(k)}=0$
5	57.8	49.2	50.7	49.7	42.7	37.9	35.0
10	178.0	156.0	148.0	126.0	105.0	112.0	128.0
20	780.0	613.0	663.0	453.0	484.0	416.0	548.0

Table 5.3 *Example 1*

n	KMW $p^{(k)}=0$	KMW B	KMHW $p^{(k)}=0$	KMHW B	KMSW $p^{(k)}=0$	KMSW B	HS $p^{(k)}=0$
5	31.4	26.6	28.4	23.9	20.9	18.8	17.4
10	135.0	100.0	113.0	87.1	91.0	66.7	93.8
20	929.0	502.0	732.0	411.0	468.0	351.0	598.0

Table 5.4 *Example 2*

n	KMW $p^{(k)}=0$	KMW B	KMHW $p^{(k)}=0$	KMHW B	KMSW $p^{(k)}=0$	KMSW B	HS $p^{(k)}=0$
5	34.2	28.8	31.0	23.8	19.8	21.0	17.4
10	147.0	96.8	121.0	85.7	89.6	81.5	103.0
20	967.0	517.0	814.0	412.0	580.0	356.0	611.0

Table 5.5 *Example 4*

n	KMW $p^{(k)}=0$	KMW B	KMHW $p^{(k)}=0$	KMHW B	KMSW $p^{(k)}=0$	KMSW B	HS $p^{(k)}=0$
5	43.8	33.0	36.4	32.4	29.7	30.2	36.2
10	137.0	91.0	127.0	88.4	89.6	85.6	107.0
20	488.0	390.0	492.0	279.0	411.0	262.0	450.0

Table 5.6 *Example 5*

n	KMW $p^{(k)}=0$	KMW B	KMHW $p^{(k)}=0$	KMHW B	KMSW $p^{(k)}=0$	KMSW B	HS $p^{(k)}=0$
9	68.5	48.8	58.8	47.0	41.4	36.7	49.9
16	301.0	207.0	247.0	199.0	172.0	129.0	201.0
25	1060.0	542.0	856.0	463.0	548.0	415.0	733.0

Table 5.7 *Example 7*

n	$s=0.5$	$s=0.6$	$s=0.7$	$s=0.8$	$s=0.9$
5	30.5	30.7	53.9	54.2	52.8
10	185.0	182.0	182.0	279.0	282.0
20	596.0	906.0	1010.0	1350.0	1370.0

Table 5.8 *HG — Example 2*

6. The *ALGLIB* package for symbolic computation

There are many instances in numerical mathematics in which the derivative of a function is required. See for example [Ral--80b]. If a package to differentiate analytically is not available then the user has to carry out the differentiation by hand, which is tedious and error-prone, or an approximation to the derivative must be used. In many cases both of these are unacceptable. Many packages to carry out symbolic manipulations are currently available, MACSYMA [Bog--77a] and REDUCE 2 [Hea--73a] being among the best known. However these packages are to a large degree 'isolated' and are difficult to access from within programs written in a language of the user's choice. An application in which a Fortran program is interfaced with REDUCE 2 has been described by Watanabe [Wat--83a].

In this chapter a library (*ALGLIB*) of procedures which perform analytic differentiation and other simple symbolic manipulations is described. The *ALGLIB* package has been written in a pseudo-code, described in Appendix C, whose meaning should be clear to a programmer in most high-level languages. The package may be implemented in the high-level language of the user's choice. Ideas on how the library of procedures might be embedded in a compiler are given. Some applications of the *ALGLIB* package in optimization and nonlinear equation solving are given

in Chapter 7.

6.1 Factorable functions

The following definitions are useful subsequently.

Definition 6.1 : Let X be a given set (e.g. R or $I(R)$), and let $+ : X \times X \to X$ and $\cdot : X \times X \to X$ be binary addition and multiplication respectively. A function $f : X^n \to X$ is a factorable function of the variables $x_i \in X$ $(i = 1, \ldots, n)$ if and only if it can be represented as the last in a finite sequence of functions $\{f_j\}$ which are defined as follows.

$$f_j(x) = x_j \qquad (j = 1, \ldots, n). \tag{6.1}$$

For $j > n$, $f_j(x)$ is of the form

$$f_k(x) + f_l(x) \qquad (k, l < j), \tag{6.2}$$

or of the form

$$f_k(x) \cdot f_l(x) \qquad (k, l < j), \tag{6.3}$$

or of the form

$$\mathrm{T}[f_k(x)] \qquad (k < j), \tag{6.4}$$

where $\mathrm{T} : X \to X$ is a given function. □

Since for the purpose of this thesis, only functions which may be represented on a computer will be considered, a more restricted class of functions, which is a subset of the set of factorable functions will be defined.

Definition 6.2 : Let X be a given set the elements of which may be represented in a computer, and let $+ : X \times X \to X$, $- : X \times X \to X$, $* : X \times X \to X$ and $/ : X \times X \to X$, be addition, subtraction, multiplication and division operators respectively. A function $f : X^n \to X$ is a computable factorable function of the variables $x_i \in X \quad (i = 1, \ldots, n)$ if and only if it can be represented as the last in a finite sequence of functions $\{f_j\}$ which are defined as follows.

$$f_j(x) = x_j \qquad (j = 1, \ldots, n). \tag{6.5}$$

For $j > n$, $f_j(x)$ has one of the following forms.

$$f_k(x) + f_l(x) \qquad (k, l < j), \tag{6.6}$$

$$f_k(x) - f_l(x) \qquad (k, l < j), \tag{6.7}$$

$$f_k(x) * f_l(x) \qquad (k, l < j), \tag{6.8}$$

$$f_k(x) / f_l(x) \qquad (k, l < j), \tag{6.9}$$

or

$$\mathrm{T}[f_k(x)] \qquad (k < j), \tag{6.10}$$

where $\mathrm{T}[\cdot] \in \mathbb{F} = \{-(\cdot), \mathrm{sqrt}(\cdot), \exp(\cdot), \ln(\cdot), \cos(\cdot), \sin(\cdot), \mathrm{atan}(\cdot), |\cdot|, (\cdot)^m$, where m is an integer $\}$, and where the functions in the set $\mathbb{F}$ are assumed to be defined on X . □

The set $\mathbb{F}$ may be extended to include any other function from X to X which may be evaluated in a particular computing environment, provided that the function is differentiable, and that the derivative of the function is itself a computable factorable function. Defintion 6.2 is illustrated by the following example.

Example 6.1 : The function $f : R^3 \to R^1$ defined by

$$f(x_1, x_2, x_3) = \cos(x_1 + x_2 * x_3)$$

is a computable factorable function since we may write

$$f_1(x) = x_1,$$

$$f_2(x) = x_2,$$

$$f_3(x) = x_3,$$

$$f_4(x) = x_2 * x_3 = f_2(x) * f_3(x),$$

$$f_5(x) = x_1 + x_2 * x_3 = f_1(x) + f_4(x),$$

$$f_6(x) = \cos(x_1 + x_2 * x_3) = \cos(f_5(x)).$$

Clearly f is equal to the last in a finite sequence of functions which satisfies the conditions of the definition. □

Owing to the nature of the differentiation operator, the partial derivative of a computable factorable function with respect to any of the variables is itself a computable factorable function. Much work has been done on computer-generated analytic derivatives of factorable functions. See, for example the work by Rall [Ral--69a], [Ral--81a], Sisser [Sis--82a], [Sis--82b], [Sis--82c], Pugh [Pug--72a] and McCormick [McC--83a].

6.2 Data structures

Given an expression which defines a computable factorable function, *ALGLIB* generates the sequence of functions $\{f_j\}$ which make up its factorable form, and stores this sequence efficiently. The function, once stored in this way may then be differentiated, evaluated, output as a string, or composed or combined with other functions which are similarly stored. In this section the data structures which are

used to store the finite sequence $\{f_j(x)\}$ are described. Any term in the sequence is one of the following:

(i) a constant;

(ii) a variable (*i.e.* Of the form (6.5));

(iii) a binary term (*i.e.* Of one of the forms (6.6)–(6.9));

(iv) a unary term (*i.e.* Of the form (6.10)).

To store a constant or a variable we need only store its name and its current value. Storing unary terms and binary terms is slightly more complicated. A unary term contains an argument and an operator, where the argument is another term in the sequence. We could therefore store a unary term in a data structure consisting of a string and a pointer; the string represents the unary operator and the pointer points to the argument. Similarly a binary term could be stored in a structure composed of the operator, in a string, and pointers to each of the sub-terms. This gives rise to a binary tree (or, more correctly, an acyclic graph), each node of which represents a term in the sequence $\{f_j\}$ and each of the leaf nodes of which is a constant or a variable. The head of the tree represents the last term in the sequence $\{f_j\}$. The function of Example 2.1 is represented by the tree structure shown in Figure 6.1.

In order to avoid storing several representations of the same object, which may occur as a result of generating several trees which contain the same term, we require a simple and efficient method for checking a new node against those which have been already created.

One approach is to link the constants together into one ordered linked list, the unary terms together into a second ordered list, and the binary nodes into a third.

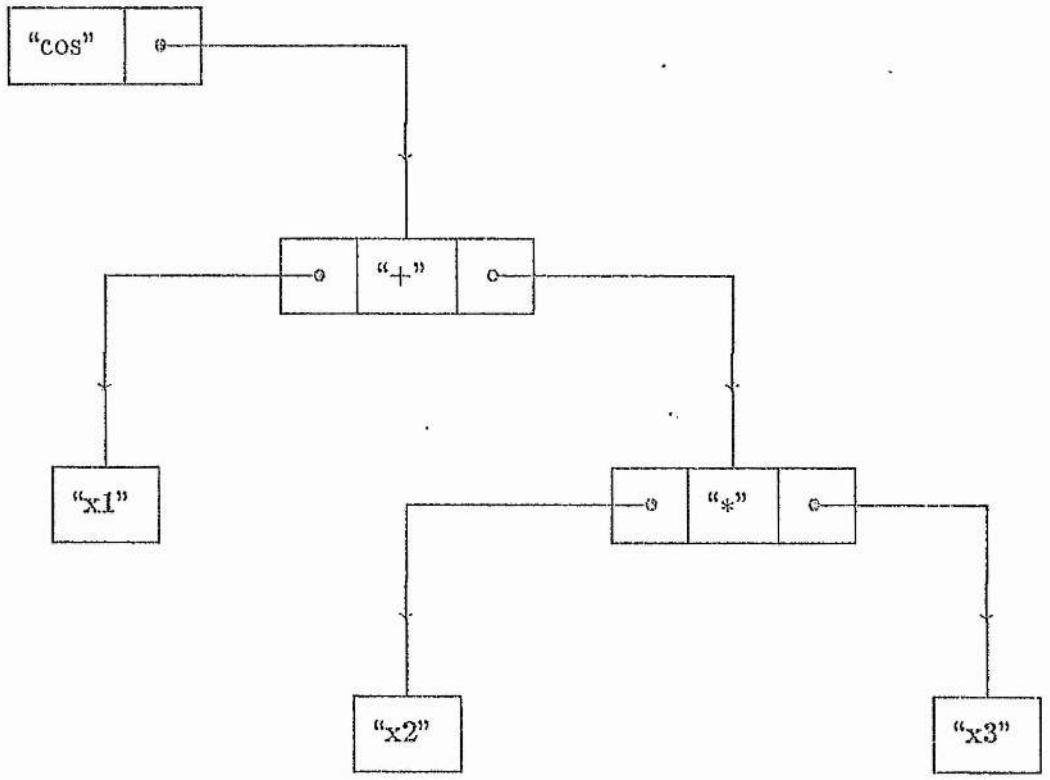


Figure 6.1

Thus when a new node representing a constant is about to be created, the constant is checked against the constants in the linked list, and if a duplicate is found then the new node is not created, but a pointer to the old representation is used. Otherwise the new node is created and is added to the linked list so as to preserve the ordering. A similar process is used for unary and binary terms. It is clear that other approaches which involve a more efficient search procedure would be desirable. However any alternative approach tends to complicate the other processes of the package and will

not be dealt with here. In order to maintain the linked lists a linking field must be introduced into the data structures for constants, unary terms, and binary terms. An index field is also included in these data structures and the data structure for variables to facilitate the ordering of the lists. Since structures to represent all the variables are created when the variables are defined we should never require a new node to represent a variable.

To keep the linked lists and the variables accessible we create an information block which is associated with each set of variables. This information block contains

(i) a vector of pointers to each of the variables;

(ii) a pointer to the list of constants;

(iii) a pointer to the list of unary terms;

(iv) a pointer to the list of binary terms.

We introduce a pointer into each structure definition which will be used to point to the information block corresponding to the variables on which the term corresponding to the structure is defined.

The preceding structures are adequate but it is desirable that the package should be able to store the value of a term so that it may be re-used automatically if the value of this term at the same point is required later. This involves introducing a field to store the value of a unary or binary term and a flag to mark whether this value is up-to-date or not into the structure definitions for unary and binary terms.

Similarly if we create a tree structure to represent the derivative of a term with respect to one of the variables, then it is desirable to keep a pointer to this tree, to avoid its being re-computed later. This involves the introduction of a vector of

pointers into the structures for unary and binary terms. Each component of the vector points to the partial derivative of the term with respect to one of the variables. A nil pointer denotes that the term's derivative with respect to this variable has not been computed.

A diagrammatic representation of the structures used in the *ALGLIB* package is given in Figure 6.2 and a description of the fields is given in Table 6.1.

Descriptor	Type	Description
name	S	The name of a variable or constant
op	S	The operator in unary or binary terms
index	Z	The index of a term
value	X	The current value of a term
root	P	A pointer to the information block
link	P	A pointer to the next term in a list
arg	P	A pointer to the argument of a unary term
left	P	A pointer to the left sub-term of a binary term
right	P	A pointer to the right sub-term of a binary term
grad	P^n	Vector of pointers to partial derivatives
known	B	Up-to-date flag
vars	P^n	Vector of pointers to the variables
cons	P	Pointer to list of constants
uns	P	Pointer to list of unary terms
bins	P	Pointer to list of binary terms

Table 6.1

6.3 An outline of basic methods

In this section we will look briefly at the approach used in the *ALGLIB* package to implement the processes of symbolic computation.

constant

name	index	value	root	link

variable

name	index	value	root

unary

arg	op	grad	index	known	value	root	link

binary

left	right	op	grad	index	known	value	root	link

root

vars	cons	bins	uns

Figure 6.2

6.3.1 Maintaining the linked lists

In Section 6.2 it was stated that linked lists of all terms currently in existence should be maintained in order to avoid repetition of like terms. In order to do this, as soon as a new term is created it should be added to the appropriate linked list. The list may be accessed through the term's root field. To make the searching of lists more efficient we introduce an ordering for terms and store terms in order within the lists. To order terms of different types we use the relation

constants < variables < unary terms < binary terms.

To order terms of the same type, if both terms are already in the linked list then we could use their index fields. If, however one of the terms has not yet been added to the linked list we cannot use the index field. We can however assume that the index fields of any of this term's sub-terms will be correct. Therefore to order unary terms we use the criteria

1. order of arguments (by type or index)
2. order of operators (alphabetic)

If the terms are equal under the first criterion, they are ordered under the second. If they are equal under both criteria then the two terms may be assumed to be equivalent.

Ordering between binary terms uses the criteria

1. order of left sub-terms (by type or index)
2. order of right sub-terms (by type or index)
3. order of operators (by ascii codes)

Constants may be ordered alphabetically on their name fields, and variables can always be ordered by their indices.

Thus to add a new term to the appropriate linked list, we chain down the list until we find a replica, in which case the new term is replaced by a pointer to the replica, or until we find the point where this new term should be inserted into the linked list.

6.3.2 Simplification

Before adding a new term to the linked list it should be simplified as much as possible. This is a difficult problem which still needs much work. Certain kinds of simplification are easy to implement and they will be described here. A binary or unary term, all of whose sub-terms are constants may be reduced to a single constant. A binary term having as one of its sub-terms the constant 0 or the constant 1 can often be simplified. A binary term both of whose sub-terms are the same may be simplified in many cases.

6.3.3 Conversion of a function to its factorable form

The problem here is to take a string as input, break it up into its separate parts, construct the sequence of functions which make up the factorable form, and return a pointer to the head of the corresponding tree. As any new term is created it should be added to the linked list of terms currently in existence. This problem is closely related to the problem of parsing expressions in compilers. We will consider here the method

of recursive descent [DavM-81a] only, since this is straightforward to implement. As Davie and Morrison point out, other methods, such as operator precedence parsing [Bor--79a], may be more efficient. We will not consider them here since recursive descent is adequate to illustrate our ideas. The syntax for expressions which has been adopted by the Author is given in Figure 6.3 in Wirth BNF. The notation of Wirth BNF is given in Table 6.2.

Symbol	Meaning
|	a choice
{ }	once or many times
[ ]	optional
⟨ ⟩	non-terminal
( )	grouping
::=	production
" "	literal

Table 6.2 *Wirth BNF*

The first phase of analysing expressions is the lexical analysis. This involves breaking up the input string into the basic symbols we are using. In our application the basic symbols are identifiers, round brackets, numbers, or one of the symbols " + " , " – " , " * " , " / " or " ^ " .. Thus in Example 6.1 the string

```
⟨term⟩                  ::=  ⟨expression1⟩
⟨expression1⟩           ::=  ⟨expression2⟩ [{(" + "|" - ") ⟨expression2⟩}]
⟨expression2⟩           ::=  [" + "|" - "] ⟨expression3⟩
⟨expression3⟩           ::=  ⟨expression4⟩ [{(" * "|" / ") ⟨expression4⟩}]
⟨expression4⟩           ::=  ⟨expression5⟩
                             [{" ^ " [(" + "|" - ")] ⟨integer⟩}]
⟨expression5⟩           ::=  ⟨unary expression⟩ |
                             ⟨variable⟩ |
                             ⟨constant⟩ |
                             "(" ⟨expression1⟩ " ) "
⟨unary expression⟩      ::=  ⟨standard function⟩ " ( " ⟨expression1⟩ " ) "
⟨standard function⟩     ::=  "sqrt"|" exp "|" ln "|" cos "|
                             "sin"|" atan "|" abs "
⟨variable⟩              ::=  ⟨identifier⟩
⟨constant⟩              ::=  ⟨standard constant⟩ | ⟨literal⟩
⟨standard constant⟩     ::=  "pi"|" epsilon "
⟨identifier⟩            ::=  ⟨letter⟩ [{⟨letter⟩ | ⟨digit⟩ |" . "}]
⟨literal⟩               ::=  ⟨integer⟩ [" . " | ⟨integer⟩]]
⟨integer⟩               ::=  {⟨digit⟩}
⟨digit⟩                 ::=  "0"|" 1 "|···|" 9 "
⟨letter⟩                ::=  "a"|" b "|···|" z "|
                             "A"|" B "|···|" Z "
```

Figure 6.3 *Expression Syntax*

" cos (x1 + x2 * x3) " should be broken up into the symbols " cos " , " (" ,

" $x1$ " , " + " , " $x2$ " , " * " " $x3$ " and ") " . This process is fairly straightforward since we can tell from the first letter of a symbol what kind of symbol it is. For example if the first character of the symbol is a digit then the symbol must be a number.

The syntax analysis is very similar to that which is described by Davie and Morrison for compiling expressions. Type checking is not required since all objects will have the same type. Instead of generating code, as in the case of compilation, we simply create the appropriate node of the tree. Table 6.3 shows the type of node generated by each syntactic construct. Table 6.3 also shows which of the forms of Definition 6.2 the function corresponding to this node has.

Implementation of the information in Figure 6.3 and tables 6.2–6.3 involves creating procedures corresponding to each of the syntactic constructs of Figure 6.3, except *digit* and *letter* which are dealt with by the lexical analyser. These procedures should check that the syntax of the symbols found in the input stream meet the requirements of the construct, and if neccessary, the appropriate node should be created as laid out in Table 6.3. This process is well documented by Davie and Morrison, so it will not be repeated here.

6.3.4 Evaluation of a function in factorable form

To evaluate a function held in the tree structure described in Section 6.2, we first put the correct value into the value field of each of the variables and then use a recursive procedure to evaluate the root node of the tree. The procedure uses a

Construct	Node	Form
term	—	—
expression1	binary term	$f_k(x) + f_l(x)$ or $f_k(x) - f_l(x)$
expression2	unary term	$+[f_k(x)]$ or $-[f_k(x)]$
expression3	binary term	$f_k(x) * f_l(x)$ or $f_k(x)/f_l(x)$
expression4	binary term	$\{f_k(x)\}^m$
expression5	—	—
unary expression	unary term	$\mathrm{T}[f_k(x)]$
standard function	—	—
variable	variable	$x_j \quad 1 \leq j \leq n$
constant	constant	—
standard constant	—	—
identifier	—	—
literal	—	—
digit	—	—
letter	—	—

Table 6.3

recursive call of itself to evaluate any sub-terms of a node. Once the values of a node's sub-terms are known, the value of the node is simple to compute, by applying the operator associated with the node to the values for the sub-terms which have been computed.

6.3.5 Partial derivatives

To compute a partial derivative of a function in a factorable form, we require only the following simple rules of differentiation, where f and g represent $f(x)$ and $g(x)$ for brevity:

$$\frac{\partial f}{\partial x_i} = 0 \qquad \text{if } f(x) \text{ is a constant;}$$

$$\frac{\partial f}{\partial x_i} = 1 \qquad \text{if } f(x) = x_i \text{ ;}$$

$$\frac{\partial f}{\partial x_i} = 0 \qquad \text{if } f(x) \text{ is a variable other than } x_i \text{ ;}$$

$$\frac{\partial}{\partial x_i}(f+g) = \frac{\partial f}{\partial x_i} + \frac{\partial g}{\partial x_i};$$

$$\frac{\partial}{\partial x_i}(f-g) = \frac{\partial f}{\partial x_i} - \frac{\partial g}{\partial x_i};$$

$$\frac{\partial}{\partial x_i}(f*g) = g\frac{\partial f}{\partial x_i} + f\frac{\partial g}{\partial x_i};$$

$$\frac{\partial}{\partial x_i}(f/g) = \frac{g\frac{\partial f}{\partial x_i} - f\frac{\partial g}{\partial x_i}}{g^2};$$

$$\frac{\partial}{\partial x_i}(f^m) = mf^{m-1}\frac{\partial f}{\partial x_i};$$

$$\frac{\partial}{\partial x_i}(\mathrm{T}[f]) = \dot{\mathrm{T}}[f]\frac{\partial f}{\partial x_i}.$$

The form of $\dot{\mathrm{T}}[f]$ for each of the unary operators is given in Table 6.4.

Since $|f|$ is not differentiable it is not included in Table 6.4. The derivative of any term may now be generated recursively. The recursion will terminate since all leaf nodes are either constants or variables.

$T[f]$	$\dot{T}[f]$
$\mathrm{sqrt}(f)$	$1/(2\,\mathrm{sqrt}(f))$
$\exp(f)$	$\exp(f)$
$\ln(f)$	$\frac{1}{f}$
$\sin(f)$	$\cos(f)$
$\cos(f)$	$-\sin(f)$
$\mathrm{atan}(f)$	$1/(1+f^2)$
$-f$	-1

Table 6.4

6.3.6 Combining functions

Combining functions is very easy. Suppose that the tree structure representing the function $f(\cdot)$ has been created. Then, in order to create the tree structure representing the function $T[f(\cdot)]$ we simply form the unary term whose operator field contains the operator T, and whose argument field contains a pointer to the tree representing $f(\cdot)$, and return a pointer to this new term. The process for binary terms is similar. If the tree structures representing the functions $f(\cdot)$ and $g(\cdot)$ have been created, and we want to create the tree representing $f \otimes g(\cdot)$, where $\otimes \in \{+, -, *, /\}$ then we create a binary term whose left and right fields point to the tree structures for $f(\cdot)$ and $g(\cdot)$ respectively, and whose operator field contains the operator $\otimes$. A pointer to this new term is returned.

6.3.7 Composing functions

Composing functions is slightly more complicated than combining them. Suppose that the tree structures representing the functions $f(\cdot) : X^n \rightarrow X$ and $g_i(\cdot) : X^n \rightarrow X \quad (i = 1, \ldots, n)$ have been created, and we wish to create the tree structure representing the function $f \circ g(\cdot) : X^n \rightarrow X$, where $f \circ g(x) = f(g(x))$. Then the tree structure of the function $f(\cdot)$ is replicated, with every occurrence of a pointer to the i^{th} variable x_i replaced by a pointer to the tree structure representing $g_i(\cdot)$.

6.4 The *ALGLIB* package

The procedures which make up the *ALGLIB* package are described in this section. The procedure headings and an example call of each procedure are given in the pseudo-code of Appendix C.

The procedure *define.variables* , which has a head of the form

$$\textbf{procedure } define.variables(variable.names \in S^n \longrightarrow P^n)$$

takes the vector of strings which are intended to make up the variable names, creates a structure of class variable corresponding to each of these names, and returns a vector of pointers, each of whose components points to one of these structures. The names are checked for uniqueness and validity. To define a set of variables whose names are stored in the vector of strings, *names* , the statement

$$variables := define.variables(names)$$

is used.

The procedure *string.to.function* , which has a head of the form

$$\textbf{procedure } string.to.function(variables \in P^n, expression \in S \longrightarrow P)$$

takes as input the set of variables on which we wish to define the function, and a string representation of the function. The tree representation of the function is created, and a pointer to it is returned. Assuming we have defined a set of variables with names " $x1$ " , " $x2$ " , " $x3$ " as described above, and that the variable *expression* , of type string, contains the character string " $cos(x1 + x2 * x3)$ " , the statement

$$f := string.to.function(variables, expression)$$

creates the tree representation of Figure 6.1 and returns a pointer to it.

The procedure *function.format* , which has a head of the form

$$\textbf{procedure } function.format(factorable \in P \longrightarrow S)$$

reverses the process of *string.to.function* , and converts a function, held in its tree representation and pointed at by *factorable* , into a string ready for output. Extensive bracketing is used to avoid ambiguity.

Once the tree representation of a function has been created, it may be evaluated by the procedure *evaluate* , which has a head of the form

$$\textbf{procedure } evaluate(factorable \in P, point \in X^n \longrightarrow X).$$

This procedure takes, as input, a pointer to the tree and the point at which the function is to be evaluated. The value of the function at the given point is returned.

For example, if the vector *point* contains the point at which the function f defined as above is to be evaluated, then the statement

$$f.of.point := evaluate(f, point)$$

will evaluate f . Two variations of this procedure are also provided to evaluate functions from X^n to X^n and functions from X^n to $M(X^n)$. The first

$$\textbf{procedure } vecevaluate(factorable \in P^n, point \in X^n \longrightarrow X^n)$$

evaluates a vector of functions at the given point while

$$\textbf{procedure } matevaluate(factorable \in M(P^n), point \in X^n \longrightarrow M(X^n))$$

evaluates the given matrix of functions at the given point.

The procedure *partial* , which has a head of the form

$$\textbf{procedure } partial(factorable, variable \in P \longrightarrow P)$$

takes a pointer to a function's tree representation and a pointer to a variable. A tree representation of the derivative of the function with respect to the variable is created, and a pointer to it is returned. Thus, the statement

$$df.by.dx1 := partial(f, variables(1))$$

where *variables*(1) points to the $\mathcal{ALGLIB}$ representation of the variable x_1 , would create the tree structure representing $\partial f/\partial x_1$, and return a pointer to it.

The procedure *function.op.function* which has a head of the form

$$\textbf{procedure } function.op.function(a \in P, op \in S, b \in P \longrightarrow P)$$

creates the tree representation of the function formed when the two functions, a and b are combined by the given binary operator op . Thus if functions f and g had been created as described previously, and the variable, *operator* contained the string " + " , then the statement

$$h := function.op.function(f, operator, g)$$

would create a tree representation of the function $h(\cdot) = f(\cdot) + g(\cdot)$. There are also two procedures which allow values to be combined with functions. They are

$$\textbf{procedure } value.op.function(a \in X, op \in S, b \in P \longrightarrow P)$$

and

$$\textbf{procedure } function.op.value(a \in P, op \in S, b \in X \longrightarrow P).$$

The procedure *op.function* which has a head of the form

$$\textbf{procedure } op.function(op \in S, arg \in P \longrightarrow P)$$

creates the tree representation of the function which is obtained when the given operator acts on the given function. If the variable *operator* contains the character string " *cos* " , then the statement

$$g := op.function(operator, f)$$

would create the tree representation of the function $g(\cdot) = cos(f(\cdot))$.

The procedure *compose* which has a head of the form

$$\textbf{procedure } compose(f \in P, g \in P^n \longrightarrow P)$$

takes as input a pointer f to the tree representation of a function from X^n to X^1, and a vector g of pointers, each of whose components points to the tree representation of one of the components of a function from X^n to X^n. A tree representation of the composition $f \circ g$ of f with g is returned. For example, the statement

$$h := compose(f, g)$$

would create the tree representation of the function $h : X^n \to X^1$ where $h(\cdot) = f(g(\cdot))$.

The *ALGLIB* package also contains other procedures which act on functions from X^n to X^n, and on functions from X^n to $M(X^n)$, where $M(X^n)$ is the set of matrices over X of order n.

The following example is of interest since it illustrates most of the facilities of the *ALGLIB* package.

Example 6.2 : The solution of sytems of equations of the form

$$Ax + d(x) + c = 0 \tag{6.11}$$

where $A \in M(R^n)$ is tridiagonal, $d : R^n \to R^n$ is a diagonal operator and $c \in R^n$, is a common problem in numerical mathematics. If $A = (a_{ij})_{n \times n}$,

$d(x) = (d_i(x_i))_{n\times 1}$ and $c = (c_i)_{n\times 1}$ then we may write (6.11) as

$$\begin{aligned}
a_{11}x_1 + a_{12}x_2 &\qquad\qquad + d_1(x_1) + c_1 = 0\\
a_{21}x_1 + a_{22}x_2 + a_{23}x_3 &\qquad\qquad + d_2(x_2) + c_2 = 0\\
\ddots \qquad\qquad \ddots &\qquad\qquad \vdots \quad \vdots \quad \vdots\\
a_{n,n-1}x_{n-1} + a_{nn}x_n &+ d_n(x_n) + c_n = 0
\end{aligned}$$

A simple re-arrangement of the first $n-1$ of these equations yields the $n-1$ equations,

$$\begin{aligned}
x_2 &= -(a_{11}x_1 + d_1(x_1) + c_1)/a_{12}\\
&= g_2(x_1)\\
x_3 &= -(a_{21}x_1 + a_{22}x_2 + d_2(x_2) + c_2)/a_{23}\\
&= -(a_{21}x_1 + a_{22}g_2(x_1) + d_2(g_2(x_1)) + c_2)/a_{23}\\
&= g_3(x_1)\\
&\vdots \qquad\qquad \vdots \qquad\qquad \vdots\\
x_n &= -(a_{n-1,n-2}x_{n-2} + a_{n-1,n-1}x_{n-1} + d_{n-1}(x_{n-1}) + c_{n-1})/a_{n-1,n}\\
&= -(a_{n-1,n-2}g_{n-2}(x_1) + a_{n-1,n-1}g_{n-1}(x_1) + d_{n-1}(g_{n-1}(x_1)) + c_{n-1})/a_{n-1,n}\\
&= g_n(x_1)
\end{aligned}$$

And the n^{th} equation becomes

$$0 = a_{n,n-1}g_{n-1}(x_1) + a_{nn}g_n(x_1) + d_n(g_n(x_1)) + c_n \qquad (6.12)$$

which is one equation in one unknown and may be solved using such methods as Newton's method. This yields x_1^* and now back-solving $x_2^* = g_2(x_1^*), \ldots, x_n^* = g_n(x_1^*)$ yields $x^* = (x_i^*)_{n\times 1}$ which is a soluton of (6.11).

Although this method would be impossible to implement without a symbolic computation package, it is quite simple to implement using *ALGLIB*. The procedure *tridiagonal.solver* of Figure 6.4 requires as input a vector of strings *variable.names* containing the variable names, the real tridiagonal matrix A, a vector of strings, *d.strings* containing expressions representing each of the functions $d_i : R^n \to R$, and the real vector c. An initial guess *starting.point* of the solution of 6.11 is also needed. The parameter *eps* is used to test for convergence of the Newton sequence and the integer *maxit* is the maximum number of Newton iterates allowed. Figure 6.5 contains a listing of the procedure *Newton* used in *tridiagonal.solver*. The procedures

$$\text{procedure } \textit{vector.pointers}(lo, hi \in Z, val \in P \longrightarrow P^n)$$

and

$$\text{procedure } \textit{vector.reals}(lo, hi \in Z, val \in R \longrightarrow R^n)$$

create a vector with lower bound *lo* and upper bound *hi*, each of whose elements is initialised to the given value *val* and return this vector as their result.

```
procedure tridiagonal.solver(variable.names ∈ S^n, A ∈ M(R^n),
    d.strings ∈ S^n, c, starting.point ∈ R^n, eps ∈ R, n, maxit ∈ Z → R^n)
  ! Firstly the ALGLIB variables are defined and pointers to the
  ! structures representing the functions d_i (i = 1, ..., n),
```

```
   ! the real number 0 and the variable x1 are created.
1. variables := define.variables(variable.names)
2. zero := string.to.function(variables, "0")
3. d := vector.pointers(1, n, zero)
4. for i = 1 to n do
   4.1. d(i) := string.to.function(variables, d.strings(i))
5. x1 := variables(1)
   ! Next the ALGLIB structures representing the functions g_i
   ! (i = 1, ..., n) described above are created.
6. g := vector.pointers(1, n, zero)
7. g(1) := x1
8. for i = 1 to n - 1 do
   begin
     8.1. di.g := compose(d(i), g)
     8.2. fi := function.op.value(g(i), "*", A(i, i))
     8.3. fi := function.op.value(fi, "+", c(i))
     8.4. fi := function.op.function(fi, "+", di.g)
     8.5. if i > 1 do
          begin
            8.5.1. a.xi := function.op.value(g(i - 1), "*", A(i, i - 1))
            8.5.2. fi := function.op.function(fi, "+", a.xi)
          end
     8.6. fi := function.op.value(fi, "/", A(i, i + 1))
     8.7. fi := op.function("-", fi)
     8.8. g(i + 1) := fi
   end
   ! Next the ALGLIB structure representing the right-hand side
   ! of (6.12) is created and pointed to by the variable f.
9. dn.g := compose(d(n), g)
10. f := function.op.value(g(n), "*", A(n, n))
11. f := function.op.function(f, "+", dn.g)
12. a.xn := function.op.value(g(n - 1), "*", A(n, n - 1))
13. f := function.op.function(f, "+", a.xn)
14. f := function.op.value(f, "+", c(n))
```

```
    ! Finally the nonlinear equation (6.11) is solved for the variable
    ! x1 and then the values of the other variables are found
    ! by back-solving.
  15. result := vector.reals(1, n, 0.0)
  16. result(1) := Newton(f, x1, starting.point(1), eps, n, maxit)
  17. for i = 2 to n do
      17.1. result(i) := evaluate(g(i), result)
return (result)
```

Figure 6.4

```
procedure Newton(f ∈ P, x ∈ P, start, eps ∈ R, n, maxit ∈ Z → R)
  1. f.prime := partial(f, x)
  2. f.over.f.prime := function.op.function(f, "/", f.prime)
  3. Newton.iterate := function.op.function(x, "−", f.over.f.prime)
  4. old := start
  5. nit := 0
  6. converged := false
  7. while nit ≤ maxit and ~ converged do
     begin
       7.1. nit := nit + 1
       7.2. vec.old := vector.reals(1, n, old)
       7.3. new := evaluate(Newton.iterate, vec.old)
       7.4. converged := (abs(new − old) ≤ eps)
       7.5. old := new
     end
return (new)
```

Figure 6.5

6.5 Embedding the library of procedures in a compiler

Instead of implementing the *ALGLIB* package as a library of external procedures, it could quite easily be incorporated into a compiler or pre-compiler. This would make the package easier to use, and programs using the package would also be more easily understood. The process would involve introducing a new data type, **function**, corresponding to the tree representation of a function. The advantage of incorporating the procedure into a compiler or pre-compiler is that several of the operations of the package may be incorporated as infix operations. For example the call,

$$f.x := evaluate(f, x)$$

could be replaced by the statement

$$f.x := f(x)$$

and a call such as

$$f := function.op.function(g, " * ", h)$$

could be replaced by the statement

$$f := g * h.$$

Of course in more sophisticated languages such as ADA, where the user may define the infix operators for different data types, this may be incorporated without the need for a pre-compiler. The advantage of these new representations should be obvious.

7. Applications of the *ALGLIB* package

The *ALGLIB* package has been implemented for real-valued functions in S-algol [ColM-82a], and for interval-valued functions in Triplex S-algol [BaiC---a], [MorC-83a] on a Vax-11/780 computer. In this chapter some applications of the *ALGLIB* package in the fields of unconstrained optimization and nonlinear equation solving are described. Results obtained from experiments using the S-algol and Triplex S-algol implementations of *ALGLIB* are given.

7.1 The packages for unconstrained optimization and nonlinear algebraic equations of Dennis and Schnabel.

A standard with which new algorithms may be compared is provided by the packages for unconstrained optimization and nonlinear equations which have been given by Dennis and Schnabel [DenS-83a]. These packages have been implemented in S-algol by Monsi [Mon---84a] and the implementation has been interfaced with the S-algol implementation of *ALGLIB* in such a way that the user can, in addition to the options for function, gradient, Jacobian and Hessian evaluation, which are provided by Dennis and Schnabel, also use *ALGLIB* . Dennis and Schnabel [DenS-83a] have given a set of test problems for use with their packages. The first three of these

problems will be used to illustrate the S-algol implementation, with and without the use of *ALGLIB*.

If *ALGLIB* is used, then the data structures corresponding to the function and to the gradient and Hessian, or to the Jacobian must be set up before numerical computation can begin. Let the CPU time which is required to set up the required data structures be t_s seconds. Let the CPU time which is required for the subsequent numerical computation be t_c seconds. Let the CPU time which is required for the computation of the solution using analytical expressions for the function, and for the gradient and Hessian, or the Jacobian, be T seconds. Table 7.1 contains results which are obtained from the package for solving nonlinear algebraic equations. The column with heading T corresponds to Newton's method with analytical Jacobian. Table 7.2 contains the results which are obtained from the package for unconstrained minimization. The column with heading T corresponds to the use of Newton's method with analytical gradient and Hessian. In tables 7.1 and 7.2, n denotes the number of variables.

Example	n	t_S	t_C	T
1	2	0.93	0.45	0.22
2	4	4.26	6.29	3.66
3	4	15.93	4.72	1.18

Table 7.1 *Nonlinear systems*

Example	n	t_S	t_C	T
1	2	2.03	5.40	2.13
2	4	11.85	7.15	4.87
3	4	77.36	22.72	2.91

Table 7.2 *Unconstrained optimization*

In tables 7.1 and 7.2 the total CPU time which is required to solve a given problem is the sum of t_s and t_c if $\mathcal{ALGLIB}$ is used; this sum should be compared with the CPU time T which is required if analytical expressions for partial derivatives are required. As may be expected, more CPU time is required if $\mathcal{ALGLIB}$ is used. One should however take into account the considerable, sometimes prohibitive, time which is required to calculate analytical expressions for partial derivatives and to correct the algebraic mistakes which seem nearly always to arise in so doing.

7.2 A modified Newton's method for minimizing unconstrained factorable functions using symbolic computation.

Sisser [Sis--82c] has described how the Hessian of a twice differentiable factorable function $f : R^n \to R$ may be expressed as sums of outer products of vectors. We have the following theorem.

Theorem 7.1 : If $f : R^n \to R$ is a twice differentiable computable factorable

function then $\exists m \in N$, $\gamma_i : R^n \to R$, $a_i : R^n \to R^n$, $b_i : R^n \to R^n$ $(i = 1, \ldots, m)$ such that

$$f''(x) = \sum_{i=1}^{m} \gamma_i(x) \left\{ a_i(x)\, b_i(x)^T + b_i(x)\, a_i(x)^T \right\}. \tag{7.1}$$

Proof : Since f is factorable, there exists a sequence of functions $\{f_p\}$ which are generated according to the rules of Definition 6.2. We will show that the second derivative of each of the functions $f_p : R^n \to R$ $(p \geq 0)$ may be expressed in the form of (7.1). Let $m^{(p)}$, $\gamma_i^{(p)}(x)$, $a_i^{(p)}(x)$, $b_i^{(p)}(x)$ $(i = 1, \ldots, m^{(p)})$ be generated as follows. If $1 \leq p \leq n$ then

$$m^{(p)} = 1,$$

$$\gamma_1^{(p)}(x) = (0)_{n \times 1} ,$$

$$a_1^{(p)}(x) = (0)_{n \times 1} ,$$

$$b_1^{(p)}(x) = (0)_{n \times 1} .$$

If $p > n$ and $f_p(x) = f_q(x) + f_r(x)$ $(q, r < p)$ then

$$m^{(p)} = m^q + m^{(r)},$$

$$\gamma_i^{(p)}(x) = \begin{cases} \gamma_i^{(q)}(x) & (i = 1, \ldots, m^{(q)}) \\ \gamma_{i-m^{(q)}}^{(r)}(x) & (i = m^{(q)} + 1, \ldots, m^{(p)}) \end{cases},$$

$$a_i^{(p)}(x) = \begin{cases} a_i^{(q)}(x) & (i = 1, \ldots, m^{(q)}) \\ a_{i-m^{(q)}}^{(r)}(x) & (i = m^{(q)} + 1, \ldots, m^{(p)}) \end{cases},$$

$$b_i^{(p)}(x) = \begin{cases} b_i^{(q)}(x) & (i = 1, \ldots, m^{(q)}) \\ b_{i-m^{(q)}}^{(r)}(x) & (i = m^{(q)} + 1, \ldots, m^{(p)}) \end{cases}.$$

If $p > n$ and $f_p(x) = f_q(x) - f_r(x)$ $(q, r < p)$ then

$$m^{(p)} = m^q + m^{(r)},$$

$$\gamma_i^{(p)}(x) = \begin{cases} \gamma_i^{(q)}(x) & (i = 1, \ldots, m^{(q)}) \\ -\gamma_{i-m^{(q)}}^{(r)}(x) & (i = m^{(q)} + 1, \ldots, m^{(p)}) \end{cases},$$

$$a_i^{(p)}(x) = \begin{cases} a_i^{(q)}(x) & (i = 1, \ldots, m^{(q)}) \\ a_{i-m^{(q)}}^{(r)}(x) & (i = m^{(q)} + 1, \ldots, m^{(p)}) \end{cases},$$

$$b_i^{(p)}(x) = \begin{cases} b_i^{(q)}(x) & (i = 1, \ldots, m^{(q)}) \\ b_{i-m^{(q)}}^{(r)}(x) & (i = m^{(q)} + 1, \ldots, m^{(p)}) \end{cases}.$$

If $p > n$ and $f_p(x) = f_q(x) * f_r(x)$ $(q, r < p)$ then

$$m^{(p)} = m^q + m^{(r)} + 1,$$

$$\gamma_i^{(p)}(x) = \begin{cases} \gamma_i^{(q)}(x) * f_r(x) & (i = 1, \ldots, m^{(q)}) \\ \gamma_{i-m^{(q)}}^{(r)}(x) * f_q(x) & (i = m^{(q)} + 1, \ldots, m^{(p)} - 1), \\ 1 & (i = m^{(p)}) \end{cases}$$

$$a_i^{(p)}(x) = \begin{cases} a_i^{(q)}(x) & (i = 1, \ldots, m^{(q)}) \\ a_{i-m^{(q)}}^{(r)}(x) & (i = m^{(q)} + 1, \ldots, m^{(p)} - 1), \\ f_q'(x) & (i = m^{(p)}) \end{cases}$$

$$b_i^{(p)}(x) = \begin{cases} b_i^{(q)}(x) & (i = 1, \ldots, m^{(q)}) \\ b_{i-m^{(q)}}^{(r)}(x) & (i = m^{(q)} + 1, \ldots, m^{(p)} - 1). \\ f_r'(x) & (i = m^{(p)}) \end{cases}$$

If $p > n$ and $f_p(x) = f_q(x)/f_r(x)$ $(q, r < p)$ then

$$m^{(p)} = m^q + m^{(r)} + 2,$$

$$\gamma_i^{(p)}(x) = \begin{cases} \gamma_i^{(q)}(x)/f_r(x) & (i = 1, \ldots, m^{(q)}) \\ -f_q(x)\gamma_{i-m^{(q)}}^{(r)}(x)/\{f_r(x)\}^2 & (i = m^{(q)}+1, \ldots, m^{(p)}-2) \\ -1/\{f_r(x)\}^2 & (i = m^{(p)}-1) \\ f^{(q)}(x)/\{f_r(x)\}^3 & (i = m^{(p)}) \end{cases},$$

$$a_i^{(p)}(x) = \begin{cases} a_i^{(q)}(x) & (i = 1, \ldots, m^{(q)}) \\ a_{i-m^{(q)}}^{(r)}(x) & (i = m^{(q)}+1, \ldots, m^{(p)}-2) \\ f_r'(x) & (i = m^{(p)}-1) \\ f_r'(x) & (i = m^{(p)}) \end{cases},$$

$$b_i^{(p)}(x) = \begin{cases} b_i^{(q)}(x) & (i = 1, \ldots, m^{(q)}) \\ b_{i-m^{(q)}}^{(r)}(x) & (i = m^{(q)}+1, \ldots, m^{(p)}-2) \\ f_q'(x) & (i = m^{(p)}-1) \\ f_r'(x) & (i = m^{(p)}) \end{cases}.$$

If $p > n$ and $f_p(x) = \mathrm{T}[f_q(x)]$ $(q < p)$ then

$$m^{(p)} = m^q + 1,$$

$$\gamma_i^{(p)}(x) = \begin{cases} \gamma_i^{(q)}(x) * \dot{\mathrm{T}}[f_q(x)] & (i = 1, \ldots, m^{(q)}) \\ \frac{1}{2}\ddot{\mathrm{T}}[f_q(x)] & (i = m^{(p)}) \end{cases},$$

$$a_i^{(p)}(x) = \begin{cases} a_i^{(q)}(x) & (i = 1, \ldots, m^{(q)}) \\ f_q'(x) & (i = m^{(p)}) \end{cases},$$

$$b_i^{(p)}(x) = \begin{cases} b_i^{(q)}(x) & (i = 1, \ldots, m^{(q)}) \\ f_q'(x) & (i = m^{(p)}) \end{cases}.$$

A simple inductive argument now shows that

$$f_p''(x) = \sum_{i=1}^{m^{(p)}} \gamma_i^{(p)}(x) \left\{ a_i^{(p)}(x)\, b_i^{(p)}(x)^T + b_i^{(p)}(x)\, a_i^{(p)}(x)^T \right\} \qquad (p \geq 0),$$

and the theorem is proved. □

As an example of the preceding ideas, consider again the function of Example 6.1. The dyads for this function are given, with $m = 2$, by

$$\gamma_1(x_1, x_2, x_3) = -\sin(x_1 + x_2 * x3),$$
$$\gamma_2(x_1, x_2, x_3) = -\frac{1}{2}\cos(x_1 + x_2 * x3),$$

$$a_{11}(x_1, x_2, x_3) = 0,$$
$$a_{12}(x_1, x_2, x_3) = 1,$$
$$a_{13}(x_1, x_2, x_3) = 0,$$

$$a_{21}(x_1, x_2, x_3) = 1,$$
$$a_{22}(x_1, x_2, x_3) = x_3,$$
$$a_{23}(x_1, x_2, x_3) = x_2,$$

$$b_{11}(x_1, x_2, x_3) = 0,$$
$$b_{12}(x_1, x_2, x_3) = 0,$$
$$b_{13}(x_1, x_2, x_3) = 1,$$

$$b_{21}(x_1, x_2, x_3) = 1,$$

$$b_{22}(x_1, x_2, x_3) = x_3,$$

$$b_{23}(x_1, x_2, x_3) = x_2.$$

The procedure to generate the dyads which make up the Hessian matrix has been implemented using *ALGLIB*, and Monsi [Mon--85a] has used this implementation to reproduce the results given by Sisser [Sis--82c] of a modified Newton's method for minimizing factorable functions. Monsi has also produced improved minimization algorithms which exploit the outer product form of the Hessian matrix.

7.3 Moore and Jones search

In Chapter 5 the search procedure of Moore and Jones is described. The Triplex S-algol implementation of *ALGLIB* has been interfaced with a Moore-Jones search procedure in such a way that either analytic expressions for the function and Jacobian, or *ALGLIB*, can be used.

The search procedure has been used to search for the unique zero of the function described in Appendix B, Example 3 with $n = 3$ in a 3-cube centred at the origin and of radius 10, both with and without the use of *ALGLIB*. *ALGLIB* requires 11.24 seconds of CPU time to set up its internal representations of the functions and its derivative. The search requires 20 bisections, 81 function evaluations, and 34 Jacobian evaluations, both using and not using *ALGLIB*. The search requires 153 seconds of CPU time to isolate the zero of the functions in a box $\underline{x}^*$ such that

$$\|w(\underline{x}^*)\| < 10^{-10},$$

when $\mathcal{ALGLIB}$ is used. When $\mathcal{ALGLIB}$ is not used 147 seconds of CPU time are required. The slight difference in CPU time is more than offset by the convenience and the reliability of using the $\mathcal{ALGLIB}$ package.

7.4 An interval form of Brown's method

In his thesis [Bro--66a] Brown describes a Newton-like method for the solution of systems of nonlinear algebraic equations which can be very effective. Further results concerning this method are given by Brown and Conté [BroC-67a], and Dennis and Brown [DenB-71a]. An algol implementation of the algorithm is described by Brown [Bro--67a]. In this section an interval form of Brown's method is presented.

7.4.1 The 3×3 case

Brown's algorithm is a modified Newton algorithm, based on Gaussian elimination. In Brown's algorithm, the most recent information is always used at each step in the elimination process. In order to clarify the ideas which underly the interval form of Brown's method the case $n = 3$ will be examined in detail. The general case is considered subsequently.

Let $f : D \subseteq R^3 \to R^3$ be a given mapping with $f \in C^1(\hat{D})$ where $\hat{D} \subset D$ is an open convex set. Let $\underline{f} : I(\hat{D}) \to I(R^3)$ and $\underline{f}' : I(\hat{D}) \to I(M(R^3))$ be inclusion monotonic interval extensions of $f : \hat{D} \to R^3$ and $f' : \hat{D} \to M(R^3)$ respectively. Let $\hat{\underline{x}} \in I(\hat{D})$ be given and suppose that f has a unique zero

$x^* \in \underline{\hat{x}}$. Let $\hat{x} = \mathrm{m}(\underline{\hat{x}})$. Then by Lemma 2.9, $\exists \theta_1 \in [0,1]$ such that

$$\begin{aligned} 0 &= f_1(x^*) \\ &= f_1(\hat{x}) + \sum_{j=1}^{3} \partial_j f_1({}^1\xi)(x_j^* - \hat{x}_j) \end{aligned} \tag{7.2}$$

where ${}^1\xi = \hat{x} + \theta_1(x^* - \hat{x})$. Suppose that $0 \notin \partial_1 \underline{f}_1(\underline{\hat{x}})$. Then $0 \neq \partial_1 f_1({}^1\xi)$ so by (7.2)

$$x_1^* = \hat{x}_1 - \Big\{ \sum_{j=2}^{3} \partial_j f_1({}^1\xi)(x_j^* - \hat{x}_j) + f_1(\hat{x}) \Big\} \Big/ \partial_1 f_1({}^1\xi) \tag{7.3}$$

$$\in \hat{x}_1 - \Big\{ \sum_{j=2}^{3} \partial_j \underline{f}_1(\underline{\hat{x}})(\underline{\hat{x}}_j - \hat{x}_j) + \underline{f}_1(\hat{x}) \Big\} \Big/ \partial_1 \underline{f}_1(\underline{\hat{x}}). \tag{7.4}$$

Define

$$y_1(x_2, x_3) = \hat{x}_1 - \Big\{ \sum_{j=2}^{3} \partial_j f_1({}^1\xi)(x_j - \hat{x}_j) + f_1(\hat{x}) \Big\} \Big/ \partial_1 f_1({}^1\xi) \tag{7.5}$$

and

$$\underline{y}_1(\underline{x}_2, \underline{x}_3) = \hat{x}_1 - \Big\{ \sum_{j=2}^{3} \partial_j \underline{f}_1(\underline{\hat{x}})(\underline{x}_j - \hat{x}_j) + \underline{f}_1(\hat{x}) \Big\} \Big/ \partial_1 \underline{f}_1(\underline{\hat{x}}). \tag{7.6}$$

Then by (7.3) and (7.4)

$$\begin{aligned} x_1^* &= y_1(x_2^*, x_3^*) \\ &\in \underline{y}_1(\underline{\hat{x}}_2, \underline{\hat{x}}_3). \end{aligned} \tag{7.7}$$

By Lemma 2.9 $\exists \theta_2 \in [0,1]$ such that

$$\begin{aligned} 0 &= f_2(x_1^*, x_2^*, x_3^*) \\ &= f_2(y_1(x_2^*, x_3^*), x_2^*, x_3^*) \\ &= f_2(y_1(\hat{x}_2, \hat{x}_3), \hat{x}_2, \hat{x}_3) + \sum_{j=2}^{3} \partial_j f_2(y_1({}^2\xi_2, {}^2\xi_3), {}^2\xi_2, {}^2\xi_3)(x_j^* - \hat{x}_j) \end{aligned} \tag{7.8}$$

where ${}^2\xi_k = \hat{x}_k + \theta_2(x_k^* - \hat{x}_k)$ $(k = 2,3)$. Now suppose that $0 \notin \partial_2 \underline{f}_2(\underline{y}_1(\underline{\hat{x}}_2, \underline{\hat{x}}_3), \underline{\hat{x}}_2, \underline{\hat{x}}_3)$ and define

$$y_2(x_3) = \hat{x}_2 - \Big\{\partial_3 f_2(y_1({}^2\xi_2, {}^2\xi_3), {}^2\xi_2, {}^2\xi_3)(x_3 - \hat{x}_3) + f_2(y_1(\hat{x}_2, \hat{x}_3), \hat{x}_2, \hat{x}_3)\Big\} \Big/ \partial_2 f_2(y_1({}^2\xi_2, {}^2\xi_3), {}^2\xi_2, {}^2\xi_3) \quad (7.9)$$

and

$$\underline{y}_2(\underline{x}_3) = \hat{x}_2 - \Big\{\partial_3 \underline{f}_2(\underline{y}_1(\underline{\hat{x}}_2, \underline{\hat{x}}_3), \underline{\hat{x}}_2, \underline{\hat{x}}_3)(\underline{x}_3 - \hat{x}_3) + \underline{f}_2(\underline{y}_1(\hat{x}_2, \hat{x}_3), \hat{x}_2, \hat{x}_3)\Big\} \Big/ \partial_2 \underline{f}_2(\underline{y}_1(\underline{\hat{x}}_2, \underline{\hat{x}}_3), \underline{\hat{x}}_2, \underline{\hat{x}}_3) \quad (7.10)$$

Then by (7.8), (7.9) and (7.10)

$$\begin{aligned} x_2^* &= y_2(x_3^*) \\ &\in \underline{y}_2(\underline{\hat{x}}_3). \end{aligned} \quad (7.11)$$

By Lemma 2.9 $\exists \theta_3 \in [0,1]$ such that

$$\begin{aligned} 0 &= f_3(x_1^*, x_2^*, x_3^*) \\ &= f_3(y_1(x_2^*, x_3^*), x_2^*, x_3^*) \\ &= f_3(y_1(y_2(x_3^*), x_3^*), y_2(x_3^*), x_3^*) \\ &= f_3(y_1(y_2(\hat{x}_3), \hat{x}_3), y_2(\hat{x}_3), \hat{x}_3) \\ &\quad + \partial_3 f_3(y_1(y_2({}^3\xi_3), {}^3\xi_3), y_2({}^3\xi_3), {}^3\xi_3)(x_3^* - \hat{x}_3) \end{aligned} \quad (7.12)$$

where ${}^3\xi_3 = \hat{x}_3 + \theta_3(x_3^* - \hat{x}_3)$. Suppose that $0 \notin \partial_3 \underline{f}_3(\underline{y}_1(\underline{y}_2(\underline{\hat{x}}_3), \underline{\hat{x}}_3), \underline{y}_2(\underline{\hat{x}}_3), \underline{\hat{x}}_3)$ and define

$$y_3 = \hat{x}_3 - f_3(y_1(y_2(\hat{x}_3), \hat{x}_3), y_2(\hat{x}_3), \hat{x}_3) / \partial_3 f_3(y_1(y_2({}^3\xi_3), {}^3\xi_3), y_2({}^3\xi_3), {}^3\xi_3) \quad (7.13)$$

and

$$\underline{y}_3 = \hat{x}_3 - \underline{f}_3(\underline{y}_1(\underline{y}_2(\hat{x}_3), \hat{x}_3), \underline{y}_2(\hat{x}_3), \hat{x}_3) / \partial_3 \underline{f}_3(\underline{y}_1(\underline{y}_2(\underline{\hat{x}}_3), \underline{\hat{x}}_3), \underline{y}_2(\underline{\hat{x}}_3), \underline{\hat{x}}_3). \quad (7.14)$$

Then by (7.12), (7.13) and (7.14),

$$x_3^* = y_3$$

$$\in \underline{y}_3.$$

Therefore, if $\exists x^* \in \hat{\underline{x}}$ such that $f(x) = 0$, then $x_1^* \in \underline{y}_1(\hat{\underline{x}}_2, \hat{\underline{x}}_3)$, $x_2^* \in \underline{y}_2(\hat{\underline{x}}_3)$ and $x_3^* \in \underline{y}_3$, possibly allowing us to bound the solution more closely. In order to generalize these ideas we must introduce the following notation, which is consistent with the notation used for the 3×3 case which has been laid out explicitly in this section.

7.4.2 Notation

Let $f : D \subseteq R^n \to R^n$ be a given mapping with $f \in C^1(\hat{D})$ where $\hat{D} \subset D$ is an open convex set. Let $\underline{f} : I(\hat{D}) \to I(R^n)$ and $\underline{f}' : I(\hat{D}) \to I(M(R^n))$ be inclusion monotonic interval extensions of $f : \hat{D} \to R^n$ and $f' : \hat{D} \to M(R^n)$ respectively. Let $\hat{\underline{x}} \in I(\hat{D})$, $\hat{x} \in \hat{\underline{x}}$ be given and let ${}^i\xi_j \in \underline{x}_j$ $(j = i, \ldots, n)$ $(i = 1, \ldots, n)$ be arbitrary. Note that subsequently, as throughout this thesis, if a summation's lower bound exceeds its upper bound, then the sum should be assumed to be void.

For $m = 0, \ldots, n$ let ${}^m a : R^{n-m} \to R^n$ be defined by

$${}^m a_i(x_{m+1}, \ldots, x_n) = \begin{cases} y_i({}^m_i\alpha(x_{m+1}, \ldots, x_n)) & (1 \le i \le m) \\ x_i & (m+1 \le i \le n) \end{cases}, \qquad (7.15)$$

where for $i = 1, \ldots, m$, ${}^m_i\alpha : R^{n-m} \to R^{n-i}$ is defined by

$${}^m_i\alpha_j(x_{m+1}, \ldots, x_n) = {}^m a_j(x_{m+1}, \ldots, x_n). \quad (i+1 \le j \le n); \qquad (7.16)$$

that is

$$ {}^{m}_{i}\alpha_j(x_{m+1},\ldots,x_n) = \begin{cases} y_j({}^{m}_{j}\alpha(x_{m+1},\ldots,x_n)) & (i+1 \le j \le m) \\ x_j & (m+1 \le j \le n) \end{cases}. \tag{7.17} $$

For $i = 1,\ldots,n$, $y_i : R^{n-i} \to R$ is defined by

$$ y_i(x_{i+1},\ldots,x_n) = \hat{x}_i - \Big\{ \sum_{j=i+1}^{n} \partial_j g_i({}^{i}\xi_i,\ldots,{}^{i}\xi_n)(x_j - \hat{x}_j) $$

$$ + g_i(\hat{x}_i,\ldots,\hat{x}_n) \Big\} \Big/ \partial_i g_i({}^{i}\xi_i,\ldots,{}^{i}\xi_n), \tag{7.18} $$

with $g_i : R^{n-i+1} \to R$ $(i = 1,\ldots,n)$ defined by

$$ g_i(x_i,\ldots,x_n) = f_i({}^{i-1}a(x_i,\ldots,x_n)). \tag{7.19} $$

Note 7.1 :

$$ {}^{0}a_i(x_1,\ldots,x_n) = x_i \qquad (i = 1,\ldots,n), \tag{7.20} $$

so

$$ g_1(x_1,\ldots,x_n) = f_1(x_1,\ldots,x_n). \ \square \tag{7.21} $$

Note 7.2 :

$$ {}^{i}_{i}\alpha_j(x_{i+1},\ldots,x_n) = x_j \qquad (j = i+1,\ldots,n), \tag{7.22} $$

so

$$ y_i({}^{i}_{i}\alpha(x_{i+1},\ldots,x_n)) = y_i(x_{i+1},\ldots,x_n). \ \square \tag{7.23} $$

Now, for $i = 1,\ldots,n$, and for $j = i,\ldots,n$, by the chain rule

$$ \partial_j g_i(x_i,\ldots,x_n) = \mathrm{D}_j f_i({}^{i-1}a(x_i,\ldots,x_n)) $$

$$ + \sum_{k=1}^{i-1} \mathrm{D}_k f_i({}^{i-1}a(x_i,\ldots,x_n)) \partial_j y_k({}^{i-1}_{k}\alpha(x_i,\ldots,x_n)) \tag{7.24} $$

in which, for $k = i-1,\ldots,1$,

$$\partial_j y_k({}^{i-1}_{k}\alpha(x_i,\ldots,x_n)) = \mathrm{D}_j y_k({}^{i-1}_{k}\alpha(x_i,\ldots,x_n)) + \sum_{l=k+1}^{i-1} \mathrm{D}_l y_k({}^{i-1}_{k}\alpha(x_i,\ldots,x_n))\partial_j y_l({}^{i-1}_{l}\alpha(x_i,\ldots,x_n)) \qquad (7.25)$$

in which, for $l = k+1,\ldots,n$,

$$\mathrm{D}_l y_k({}^{i-1}_{k}\alpha(x_i,\ldots,x_n)) = -\partial_l g_k({}^{k}\xi_k,\ldots,{}^{k}\xi_n)\Big/\partial_k g_k({}^{k}\xi_k,\ldots,{}^{k}\xi_n). \qquad (7.26)$$

Note 7.3 : From (7.25), for $j = i,\ldots,n$,

$$\partial_j y_{i-1}({}^{i-1}_{i-1}\alpha(x_i,\ldots,x_n)) = \mathrm{D}_j y_{i-1}(x_i,\ldots,x_n) = -\frac{\partial_j g_{i-1}({}^{i-1}\xi_{i-1},\ldots,{}^{i-1}\xi_n)}{\partial_{i-1} g_{i-1}({}^{i-1}\xi_{i-1},\ldots,{}^{i-1}\xi_n)}. \qquad (7.27)$$

This reflects the recursive way in which the $\partial_j y_k({}^{i-1}_{k}\alpha(x_i,\ldots,x_n))$ are determined. If $\partial_j g_k({}^{k}\xi_k,\ldots,{}^{k}\xi_n)$ $(j = k,\ldots,n)$ $(k = 1,\ldots,i-1)$, are known, then $\partial_j y_{i-1}({}^{i-1}_{i-1}\alpha(x_i,\ldots,x_n))$ may be computed from (7.27). Then $\partial_j y_{i-2}({}^{i-1}_{i-2}\alpha(x_i,\ldots,x_n))$ may be computed from (7.25) and so on until eventually $\partial_j y_k({}^{i-1}_{k}\alpha(x_i,\ldots,x_n))$ $(k = i-1,\ldots,1)$ have been computed, when $\partial_j g_i(x_i,\ldots,x_n)$ may be computed. □

For $m = 0,\ldots,n$ let ${}^{m}\underline{\mathrm{a}} : I(R^{n-m}) \to I(R^n)$ be defined by

$${}^{m}\underline{\mathrm{a}}_i(\underline{\mathrm{x}}_{m+1},\ldots,\underline{\mathrm{x}}_n) = \begin{cases} \underline{\mathrm{y}}_i({}^{m}_{i}\underline{\alpha}(\underline{\mathrm{x}}_{m+1},\ldots,\underline{\mathrm{x}}_n)) & (1 \le i \le m) \\ \underline{\mathrm{x}}_i & (m+1 \le i \le n) \end{cases}, \qquad (7.28)$$

where for $i = 1,\ldots,m$, ${}^{m}_{i}\underline{\alpha} : I(R^{n-m}) \to I(R^{n-i})$ is defined by

$${}^{m}_{i}\underline{\alpha}_j(\underline{\mathrm{x}}_{m+1},\ldots,\underline{\mathrm{x}}_n) = {}^{m}\underline{\mathrm{a}}_j(\underline{\mathrm{x}}_{m+1},\ldots,\underline{\mathrm{x}}_n) \qquad (i+1 \le j \le n). \qquad (7.29)$$

For $i = 1,\ldots,n$, $\underline{\mathrm{y}}_i : I(R^{n-i}) \to I(R)$ is defined by

$$\underline{\mathrm{y}}_i(\underline{\mathrm{x}}_{i+1},\ldots,\underline{\mathrm{x}}_n) = \hat{x}_i - \Big\{ \sum_{j=i+1}^{n} \partial_j \underline{\mathrm{g}}_i(\hat{\underline{\mathrm{x}}}_i,\ldots,\hat{\underline{\mathrm{x}}}_n)(\underline{\mathrm{x}}_j - \hat{x}_j)$$

$$+ \underline{g}_i(\hat{x}_i, \ldots, \hat{x}_n)\Big\} \Big/ \partial_i \underline{g}_i(\hat{\underline{x}}_i, \ldots, \hat{\underline{x}}_n), \tag{7.30}$$

with $\underline{g}_i : I(R^{n-i+1}) \to I(R) \quad (i = 1, \ldots, n)$ defined by

$$\underline{g}_i(\underline{x}_i, \ldots, \underline{x}_n) = \underline{f}_i({}^{i-1}\underline{a}(\underline{x}_i, \ldots, \underline{x}_n)). \tag{7.31}$$

For $i = 1, \ldots, n$, and for $j = i, \ldots, n$,

$$\partial_j \underline{g}_i(\underline{x}_i, \ldots, \underline{x}_n) = \mathrm{D}_j \underline{f}_i({}^{i-1}\underline{a}(\underline{x}_i, \ldots, \underline{x}_n)) + \sum_{k=1}^{i-1} \mathrm{D}_k \underline{f}_i({}^{i-1}\underline{a}(\underline{x}_i, \ldots, \underline{x}_n)) \partial_j \underline{y}_k({}^{i-1}_{k}\underline{\alpha}(\underline{x}_i, \ldots, \underline{x}_n)) \tag{7.32}$$

in which, for $k = i - 1, \ldots, 1$,

$$\partial_j \underline{y}_k({}^{i-1}_{k}\underline{\alpha}(\underline{x}_i, \ldots, \underline{x}_n)) = \mathrm{D}_j \underline{y}_k({}^{i-1}_{k}\underline{\alpha}(\underline{x}_i, \ldots, \underline{x}_n)) + \sum_{l=k+1}^{i-1} \mathrm{D}_l y_k({}^{i-1}_{k}\alpha(\underline{x}_i, \ldots, \underline{x}_n)) \partial_j \underline{y}_l({}^{i-1}_{l}\underline{\alpha}(\underline{x}_i, \ldots, \underline{x}_n)) \tag{7.33}$$

in which, for $l = k + 1, \ldots, n$,

$$\mathrm{D}_l \underline{y}_k({}^{i-1}_{k}\underline{\alpha}(\underline{x}_i, \ldots, \underline{x}_n)) = -\partial_l \underline{g}_k(\hat{\underline{x}}_k, \ldots, \hat{\underline{x}}_n) \Big/ \partial_k \underline{g}_k(\hat{\underline{x}}_k, \ldots, \hat{\underline{x}}_n). \tag{7.34}$$

7.4.3 The algorithm IBSW

The algorithm IBSW is an interval version of the algorithm described by Brown [Bro--66a]. The algorithm given in this section should be modified in certain obvious ways to obtain a practical implementation. For example, a convergence test and a pivoting strategy should be included; these have been omitted for clarity. Other modifications, such as the use of an inner iteration, may be introduced. Let $\underline{x}^{(0)} \in I(R^n)$ be given. Then the algorithm IBSW for bounding a zero of $f : R^n \to R^n$ is as follows.

Algorithm 7.1 (IBSW)

1. $k := 0$

2. while true do

   2.1. $\underline{z}^{(k)} := \underline{x}^{(k)}$

   2.2. $x^{(k)} := \mathrm{m}\left(\underline{x}^{(k)}\right)$

   2.3. $\underline{v}^{(k)} := x^{(k)}$

   2.4. for $i = 1$ to n do

      2.4.1. for $j = 1$ to n do

         2.4.1.1. $\underline{F}_{ij}^{(k)} := \mathrm{D}_j \underline{f}_i\left(\underline{z}^{(k)}\right)$

      2.4.2. for $j = i$ to n do

         2.4.2.1. for $l = i - 1$ to 1 by -1 do

            2.4.2.1.1. $\underline{b}_{lj}^{(k)} := -\left\{\underline{c}_{lj}^{(k)} + \sum_{m=l+1}^{i-1} \underline{c}_{lm}^{(k)}\, \underline{b}_{mj}^{(k)}\right\} \Big/ \underline{c}_{ll}^{(k)}$

         2.4.2.2. $\underline{c}_{ij}^{(k)} := \underline{F}_{ij}^{(k)} + \sum_{l=1}^{i-1} \underline{F}_{il}^{(k)}\, \underline{b}_{lj}^{(k)}$

      2.4.3. $\underline{f}_i^{(k)} := \underline{f}_i\left(\underline{v}^{(k)}\right)$

      2.4.4. $\underline{z}_i^{(k)} := x_i^{(k)} - \left\{ \sum_{l=i+1}^{n} \underline{c}_{il}^{(k)} \left(\underline{x}_l^{(k)} - x_l^{(k)}\right) + \underline{f}_i^{(k)} \right\} \Big/ \underline{c}_{ii}^{(k)}$

      2.4.5. $\underline{v}_i^{(k)} := x_i^{(k)} - \underline{f}_i^{(k)} \Big/ \underline{c}_{ii}^{(k)}$

      2.4.6. for $j = i - 1$ to 1 by -1 do

         2.4.6.1. $\underline{z}_j^{(k)} := x_j^{(k)} - \left\{ \sum_{l=j+1}^{n} \underline{c}_{jl}^{(k)} \left(\underline{z}_l^{(k)} - x_l^{(k)}\right) + \underline{f}_j^{(k)} \right\} \Big/ \underline{c}_{jj}^{(k)}$

2.4.6.2. $\underline{v}_j^{(k)} := x_j^{(k)} - \left\{ \sum_{l=j+1}^{i} \underline{c}_{jl}^{(k)} \left(\underline{v}_l^{(k)} - x_l^{(k)} \right) + \underline{f}_j^{(k)} \right\} \Big/ \underline{c}_{jj}^{(k)}$

2.5. $\underline{x}^{(k+1)} := \underline{x}^{(k)} \cap \underline{z}^{(k)}$

2.6. $k := k + 1$ □

Note 7.4 : Steps 2.4.1–2.4.2.2 set $\underline{c}_{ij}^{(k)} = \partial_j \underline{g}_i \left(\underline{x}_i^{(k)}, \ldots, \underline{x}_n^{(k)} \right) \quad (j = i, \ldots, n)$. □

Note 7.5 : Step 2.4.4 is equivalent to setting $\underline{z}_i^{(k)} = \underline{y}_i \left(\underline{x}_{i+1}^{(k)}, \ldots, \underline{x}_n^{(k)} \right)$ while step 2.4.5 is equivalent to setting $\underline{v}_i^{(k)} = \underline{y}_i \left(x_{i+1}^{(k)}, \ldots, x_n^{(k)} \right)$. □

Note 7.6 : Steps 2.4.6–2.4.6.2 are equivalent to backsolving for $\underline{z}_j^{(k)} = \underline{y}_j \left({}_j^i \underline{\alpha} \left(\underline{x}_{i+1}^{(k)}, \ldots, \underline{x}_n^{(k)} \right) \right)$ and $\underline{v}_j^{(k)} = \underline{y}_j \left({}_j^i \underline{\alpha} \left(x_{i+1}^{(k)}, \ldots, x_n^{(k)} \right) \right) \quad (j = i-1, \ldots, 1)$. □

7.4.4 Retention of the solution

In order to show that, if $x^* \in \hat{\underline{x}}$ is such that $f(x^*) = 0$, and the sequence $\left(\underline{x}^{(k)} \right)$ is generated form IBSW with $\underline{x}^{(0)} = \hat{\underline{x}}$ then $x^* \in \underline{x}^{(k)} \quad (\forall k \geq 0)$, we need to show that for $i = 1, \ldots, n$ and for $j = 1, \ldots, i$,

$$x_j^* \in \underline{y}_j \left({}_j^i \underline{\alpha} \left(\hat{\underline{x}}_{i+1}, \ldots, \hat{\underline{x}}_n \right) \right). \tag{7.35}$$

By Lemma 2.9, for $i = 1, \ldots, n$, and for $j = 1, \ldots, i$, $\exists \left({}^i\xi_i, \ldots, {}^i\xi_n \right)^T \in R^{n-i+1}$ such that if $\hat{x} \in \hat{\underline{x}}$ then

$$g_i(x_i^*, \ldots, x_n^*) = g_i(\hat{x}_i, \ldots, \hat{x}_n) + \sum_{j=i}^{n} \partial_j g_i({}^i\xi_i, \ldots, {}^i\xi_n)(x_j^* - \hat{x}_j). \tag{7.36}$$

In the remainder of this chapter we shall suppose that the ${}^i\xi_j$ are defined by (7.36), and that the ${}^i\xi_j$ are used in (7.18) to define the y_i. First we shall show that, for $i = 1, \ldots, n$, and for $j = 1, \ldots, i$,

$$x_j^* = y_j({}^i_j\alpha(x_{i+1}^*, \ldots, x_n^*)). \tag{7.37}$$

Then we shall show that for $i = 1, \ldots, n$ and for $j = 1, \ldots, i$,

$$y_j({}^i_j\alpha(x_{i+1}, \ldots, x_n)) \in \underline{y}_j({}^i_j\underline{\alpha}(\underline{x}_{i+1}, \ldots, \underline{x}_n)) \qquad (\forall x \in \underline{x}) \tag{7.38}$$

whence since $x^* \in \hat{\underline{x}}$, it follows that for $i = 1, \ldots, n$ and for $j = 1, \ldots, i$,

$$\begin{aligned} x_j^* &= y_j({}^i_j\alpha(x_{i+1}^*, \ldots, x_n^*)) \\ &\in \underline{y}_j({}^i_j\underline{\alpha}(\hat{\underline{x}}_{i+1}, \ldots, \hat{\underline{x}}_n)) \end{aligned} \tag{7.39}$$

as required.

Lemma 7.1 : If $x^*, \hat{x} \in \hat{\underline{x}}$ and $f(x^*) = 0$ then

$$x_j^* = y_j({}^i_j\alpha(x_{i+1}^*, \ldots, x_n^*)) \qquad (j = 1, \ldots, i) \quad (i = 1, \ldots, n). \tag{7.40}$$

Proof : By Note 7.1, $g_1(x_1^*, \ldots, x_n^*) = f(x_1^*, \ldots, x_n^*) = 0$, whence a simple rearrangement of (7.36) shows that (7.40) holds for $i = 1$. Assume that, for some $m \geq 2$, (7.40) holds for $i = 1, \ldots, m-1$. Then

$$\begin{aligned} g_m(x_m^*, \ldots, x_n^*) &= f_m(y_1({}^{m-1}_1\alpha(x_m^*, \ldots, x_n^*)), \\ &\qquad \ldots, y_{m-1}({}^{m-1}_{m-1}\alpha(x_m^*, \ldots, x_n^*)), x_m^*, \ldots, x_n^*) \\ &= f_m(x_1^*, \ldots, x_n^*) \\ &= 0 \end{aligned}$$

whence, by (7.36),

$$x_m^* = \hat{x}_m - \Big\{ \sum_{j=m+1}^{n} \partial_j g_m({}^m\xi_m, \ldots, {}^m\xi_n)(x_j^* - \hat{x}_j) \Big\} \Big/ \partial_m g_m({}^m\xi_m, \ldots, {}^m\xi_n)$$
$$= y_m(x_{m+1}^*, \ldots, x_n^*)$$
$$= y_m({}^m_m\alpha(x_{m+1}^*, \ldots, x_n^*)). \tag{7.41}$$

Assume that, for some $p \leq m-1$,

$$x_j^* = y_j({}^m_j\alpha(x_{m+1}^*, \ldots, x_n^*)) \qquad (p+1 \leq j \leq m). \tag{7.42}$$

Then by Note 7.2 and Note 7.5,

$$x_p^* = y_p({}^p_p\alpha(x_{p+1}^*, \ldots, x_n^*))$$
$$= y_p(x_{p+1}^*, \ldots, x_n^*)$$
$$= y_p(y_{p+1}({}^m_{p+1}\alpha(x_{m+1}^*, \ldots, x_n^*)), \ldots, y_m({}^m_m\alpha(x_{m+1}^*, \ldots, x_n^*)), x_{m+1}^*, \ldots, x_n^*)$$
$$= y_p({}^m_p\alpha(x_{m+1}^*, \ldots, x_n^*)).$$

So (7.40) holds for $i = m$ by finite induction on j . Therefore by finite induction on i , (7.40) holds for $i = 1, \ldots, n$ and for $j = 1, \ldots, i$. □

Lemma 7.2 : If $\hat{x}_j \in \hat{\underline{x}}_j$ $(j = 1, \ldots, n)$ then for $i = 1, \ldots, n$ and for $j = 1, \ldots, i$,

$$y_j({}^i_j\alpha(x_{i+1}, \ldots, x_n)) \in \underline{y}_j({}^i_j\underline{\alpha}(\underline{x}_{i+1}, \ldots, \underline{x}_n)) \qquad (\forall x \in \underline{x}). \tag{7.43}$$

Proof : The proof is lengthy and uninstructive so only an outline will be given. By Note 7.1 and the inclusion monotonicity of $\underline{f}'$,

$$\partial_j g_1(x_1, \ldots, x_n) \in \partial_j \underline{g}_1(\underline{x}_1, \ldots, \underline{x}_n) \qquad (\forall x \in \underline{x}) \quad (i = 1, \ldots, n) \tag{7.44}$$

whence by (7.18) and (7.30), (7.43) holds for $i = 1$.

If (7.43) holds for $i = 1, \ldots, m-1$, and for $i = 1, \ldots, m-1$ and for $j = i, \ldots, n$,

$$\partial_j g_i(x_i, \ldots, x_n) \in \partial_j \underline{g}_i(\underline{x}_i, \ldots, \underline{x}_n) \qquad (\forall x \in \underline{x}) \tag{7.45}$$

then, for $j = m, \ldots, n$,

$$\partial_j g_m(x_m, \ldots, x_n) \in \partial_j \underline{g}_m(\underline{x}_m, \ldots, \underline{x}_n) \qquad (\forall x \in \underline{x}), \tag{7.46}$$

whence

$$y_m({}^m_m\alpha(x_{m+1}, \ldots, x_n)) \in \underline{y}_m({}^m_m\underline{\alpha}(\underline{x}_{m+1}, \ldots, \underline{x}_n)) \qquad (\forall x \in \underline{x}). \tag{7.47}$$

If (7.43) holds for $i = 1, \ldots, m-1$, then (7.45) holds for $i = 1, \ldots, m$ and if for some $p \leq m-1$, for $j = m, \ldots, p+1$,

$$y_j({}^m_j\alpha(x_{m+1}, \ldots, x_n)) \in \underline{y}_j({}^m_j\underline{\alpha}(\underline{x}_{m+1}, \ldots, \underline{x}_n)) \qquad (\forall x \in \underline{x}) \tag{7.48}$$

then

$$y_p({}^m_p\alpha(x_{m+1}, \ldots, x_n)) \in \underline{y}_p({}^m_p\underline{\alpha}(\underline{x}_{m+1}, \ldots, \underline{x}_n)) \qquad (\forall x \in \underline{x}) \tag{7.49}$$

whence by finite induction on j , (7.43) holds for $i = m$.

By finite induction on i ,

$$\partial_j g_i(x_i, \ldots, x_n) \in \partial_j \underline{g}_i(\underline{x}_i, \ldots, \underline{x}_n) \qquad (\forall x \in \underline{x}) \quad (j = i, \ldots, n) \quad (i = 1, \ldots, n)$$

and

$$y_j({}^i_j\alpha(x_{i+1}, \ldots, x_n)) \in \underline{y}_j({}^i_j\underline{\alpha}(\underline{x}_{i+1}, \ldots, \underline{x}_n))$$

$$(\forall x \in \underline{x}) \quad (j = 1, \ldots, i) \quad (i = 1, \ldots, n). \qquad \square$$

We have proved the following theorem.

Theorem 7.2 : If $\hat{x}, x^* \in \hat{\underline{x}}$, and $f(x^*) = 0$ then, for $i = 1, \ldots, n$ and for $j = 1, \ldots, i$,

$$x_j^* = y_j({}_j^i\alpha(x_{i+1}^*, \ldots, x_n^*))$$
$$\in \underline{y}_j({}_j^i\underline{\alpha}(\hat{\underline{x}}_{i+1}, \ldots, \hat{\underline{x}}_n)). \quad \square$$

By Theorem 7.2, it follows that if $\left(\underline{x}^{(k)}\right)$ is generated from IBSW and $\exists x^* \in \underline{x}^{(0)}$ such that $f(x^*) = 0$ then $x^* \in \underline{x}^{(k+1)} \in \underline{x}^{(k)}$ $(\forall k \geq 0)$.

7.4.5 Existence, uniqueness and convergence

No existence, uniqueness and convergence results have been obtained for the interval Brown method, but it appears likely that results similar to those set out in Conjecture 7.1 hold.

Conjecture 7.1 : If

$$\underline{y}_i({}_i^i\underline{\alpha}(\hat{\underline{x}}_{i+1}, \ldots, \hat{\underline{x}}_n)) \subseteq \hat{\underline{x}}_i \qquad (i = 1, \ldots, n) \tag{7.51}$$

and

$$w\left(\underline{y}_i({}_i^i\underline{\alpha}(\hat{\underline{x}}_{i+1}, \ldots, \hat{\underline{x}}_n))\right) < w(\hat{\underline{x}}_i) \qquad (i = 1, \ldots, n) \tag{7.52}$$

then $\exists x^* \in \hat{\underline{x}}$ such that x^* is the unique zero of f in $\hat{\underline{x}}$ and if the sequence $\left(\underline{x}^{(k)}\right)$ is generated from IBSW with $\underline{x}^{(0)} = \hat{\underline{x}}$ then $\underline{x}^{(k)} \to x^*$ $(k \to \infty)$. $\square$

7.4.6 Numerical results

The algorithm IBSW has been implemented in Triplex S-algol and has been interfaced with the Triplex implementation of *ALGLIB* . In this section numerical results are given which allow the interval Brown algorithm (IBSW), the interval Newton algorithm (IN), and the Krawczyk-Moore algorithm (KM) to be compared.

Table 7.3 and Table 7.4 contain the CPU times in seconds and the number of iterations required for convergence to be obtained for Example 4 with $n = 10$ and Example 10 with $n = 5$. The convergence criterion is $\|\mathrm{w}\left(\underline{\mathrm{x}}^{(k)}\right)\| < 10^{-10}$. For both examples $\underline{\mathrm{x}}^{(0)} = \left(\underline{\mathrm{x}}_i^{(0)}\right)_{n\times 1}$ where for $i = 1,\ldots,n$, $\underline{\mathrm{x}}_i^{(0)}$ has the value $[-4,4]$ and $[0.105, 1.05]$ for examples 4 and 10 respectively. Note that Example 10 is particularly well-suited to solution by the interval Brown algorithm. The algorithm IBSW will converge in one iteration from any initial box containing the solution.

Example	IBSW	IN	KM
4	375.84	329.37	2041.44
10	1.29	4.57	12.00

Table 7.3 *CPU times*

Example	IBSW	IN	KM
4	4	4	75
10	1	3	5

Table 7.4 *Iterations*

If results such as those given in Conjecture 7.1 are found to be valid, the interval Brown algorithm could be incorporated into a Moore-Jones search procedure. In Example 9, with $n = 5$ the Jacobian is singular at the solution $x^* = (1)_{5\times 1}$. This causes both the interval Newton and Krawczyk-Moore existence tests to fail, since neither the interval Newton or Krawczyk operator is defined. If however, $\underline{\hat{x}} = (\underline{\hat{x}}_i)_{5\times 1}$ where for $i = 1, \ldots, 5$, $\underline{\hat{x}}_i$ has the value $[0.96, 1.04]$, the conditions of Conjecture 7.1 are satisfied. Furthermore the algorithm IBSW with $\underline{x}^{(0)} = \underline{\hat{x}}$ converges to the solution in one iteration.

Appendix A — Notation

In this thesis R^n , $M(R^n)$, $I(R^n)$, and $I(M(R^n))$ denote the sets of real $n \times 1$ vectors, real $n \times n$ matrices, real $n \times 1$ interval vectors, and real $n \times n$ interval matrices respectively. Lower-case italic letters and upper-case italic letters are used to denote elements of R^n and $M(R^n)$ respectively. Underlined lower-case bold letters and underlined upper-case bold letters are used to denote elements of $I(R^n)$ and $I(M(R^n))$ respectively. Exceptions, such as $\underline{\mathbf{K}}$, $\underline{\mathbf{H}}$ and $\underline{\mathbf{S}}$ are defined in the text and are used in deference to common usage.

A real vector $x = (x_i)_{n\times 1}$ has i^{th} element x_i $(i = 1, \ldots, n)$ and a real matrix $A = (a_{ij})_{n\times n}$ has ij^{th} element a_{ij} $(i, j = 1, \ldots, n)$. An interval $\underline{\mathbf{x}} = [x_I, x_S]$ has infimum x_I and supremum x_S . An interval vector $\underline{\mathbf{x}} = (\underline{\mathbf{x}}_i)_{n\times 1}$ has i^{th} element $\underline{\mathbf{x}}_i = [x_{iI}, x_{iS}]$ $(i = 1, \ldots, n)$. An interval matrix $\underline{\mathbf{A}} = (\underline{\mathbf{a}}_{ij})_{n\times n}$ has ij^{th} element $\underline{\mathbf{a}}_{ij} = [a_{ijI}, a_{ijS}]$ $(i, j = 1, \ldots, n)$.

The mapping $|\cdot| : R^n \to R^n$ is defined by $|x| = (|x_i|)_{n\times 1}$. The sets R^n and $M(R^n)$ are partially ordered through $(x \leq y) \iff (x_i \leq y_i \ (i = 1, \ldots, n))$ and $(A \leq B) \iff (a_{ij} \leq b_{ij} \ (i, j = 1, \ldots, n))$ respectively.

The width, midpoint, and magnitude mappings $\mathrm{w} : I(R) \to R$, $\mathrm{m} : I(R) \to R$ and $|\cdot| : I(R) \to R$ are defined by

$$\mathrm{w}(\underline{x}) = x_S - x_I,$$

$$\mathrm{m}(\underline{x}) = (x_I + x_S)/2,$$

and

$$|\underline{x}| = \max\{|x_I|, |x_S|\}$$

respectively. The width, midpoint and magnitude mappings $\mathrm{w} : I(R^n) \to R^n$, $\mathrm{m} : I(R^n) \to R^n$ and $|\cdot| : I(R^n) \to R^n$ are defined by

$$\mathrm{w}(\underline{x}) = (\mathrm{w}(\underline{x}_i))_{n\times 1} ,$$

$$\mathrm{m}(\underline{x}) = (\mathrm{m}(\underline{x}_i))_{n\times 1} ,$$

and

$$|\underline{x}| = (|\underline{x}_i|)_{n\times 1}$$

respectively. The width, midpoint and magnitude mappings $\mathrm{w} : I(M(R^n)) \to M(R^n)$, $\mathrm{m} : I(M(R^n)) \to M(R^n)$ and $|\cdot| : I(M(R^n)) \to M(R^n)$ are defined by

$$\mathrm{w}(\underline{\mathbf{A}}) = \left(\mathrm{w}(\underline{a}_{ij})\right)_{n\times n} ,$$

$$\mathrm{m}(\underline{\mathbf{A}}) = \left(\mathrm{m}(\underline{a}_{ij})\right)_{n\times n} ,$$

and

$$|\underline{\mathbf{A}}| = \left(|\underline{a}_{ij}|\right)_{n\times n}$$

respectively.

The maximum norms $\|\cdot\| : R^n \to R$, $\|\cdot\| : M(R^n) \to R$, $\|\cdot\| : I(R^n) \to R$ and $\|\cdot\| : I(M(R^n)) \to R$, defined by

$$\|x\| = \max_{1\le i\le n}\{|x_i|\},$$

$$\|A\| = \max_{1\le i\le n}\left\{\sum_{j=1}^{n}|a_{ij}|\right\},$$

$$\|\underline{x}\| = \max_{1\le i\le n}\{|\underline{x}_i|\},$$

and

$$\|\underline{A}\| = \max_{1\le i\le n}\left\{\sum_{j=1}^{n}|\underline{a}_{ij}|\right\},$$

respectively, are used unless otherwise stated.

The mapping $\sigma_1 : I(R)\times I(R)\to R$ is defined by

$$\sigma_1(\underline{x},\underline{y}) = \max\{|x_I - y_I|, |x_S - y_S|\}.$$

The set $I(R^n)$ is a complete metric space with respect to the metric $\sigma_n : I(R^n)\times I(R^n)\to R$ defined by

$$\sigma_n(\underline{\mathbf{x}},\underline{\mathbf{y}}) = \max_{1\le i\le n}\left\{\sigma_1(\underline{x}_i,\underline{y}_i)\right\}.$$

The distance mapping $q : I(R^n)\times I(R^n)\to R^n$ is defined by

$$q(\underline{\mathbf{x}},\underline{\mathbf{y}}) = \left(\sigma_1(\underline{x}_i,\underline{y}_i)\right)_{n\times 1}.$$

A mapping $g : D\subseteq R^n\to R^n$ is isotone in $\hat{D}\subset D$ if and only if $(x,y\in\hat{D}\ \wedge\ x\le y)\ \Rightarrow\ (g(x)\le g(y))$, and is diagonal if and only if $g_i(x) = g_i(x_i)$ $(i = 1,\ldots,n)$ where $g(x) = (g_i(x))_{n\times 1}$.

Further $(\forall A\in M(R^n))$, $\rho(A)$ denotes the spectral radius of A and if $f : D\subseteq R^n\to R^n$ is m times continuously differentiable on an open set $\hat{D}\subset D$

then this is denoted by $f \in C^m(\hat{D})$. If $f : D \subseteq R^n \to R^n$ is a given mapping with $f \in C^1(D)$ then

$$D_j f_i(x_1, \ldots, x_n) = \frac{\partial}{\partial x_j} f_i(x_1, \ldots, n) \qquad (i, j = 1, \ldots, n).$$

If $\phi_k : R^n \to R \quad (k = 1, \ldots, n)$ are given mappings then for $i, j = 1, \ldots, n$,

$$\partial_j f_i(\phi_1(x_1, \ldots, x_n), \ldots, \phi_n(x_1, \ldots, x_n)) = \frac{\partial}{\partial x_j} f_i(\phi_1(x_1, \ldots, x_n), \ldots, \phi_n(x_1, \ldots, x_n)).$$

Appendix B — Example problems

This appendix contains several examples which are used to illustrate the relative effectiveness of the algorithms presented in this thesis.

Example 1 : The two-point boundary value problem

$$u''(t) = \exp(u(t)) \qquad (0 \le t \le 1), \tag{B.1}$$

$$u(0) = 0, \qquad u(1) = 0 \tag{B.2}$$

may be discretized [OrtR-69a] to yield $f(x) = 0$ where $f : R^n \to R^n$ is of the form

$$f(x) = Ax + Td(x) + c \tag{B.3}$$

in which $A \in M(R^n)$ is given by

$$A = \begin{pmatrix} 2 & -1 & & & O \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ O & & & -1 & 2 \end{pmatrix}_{n \times n}, \tag{B.4}$$

$d : R^n \to R^n$ is defined by

$$d_i(x) = h^2 \exp(x_i) \qquad (i = 1, \ldots, n), \tag{B.5}$$

where $h = \frac{1}{n+1}$, and $c \in R^n$ is given by

$$c_i = 0 \qquad (i = 1, \ldots, n). \tag{B.6}$$

Example 1 is quoted by Alefeld [Ale--72a]. □

Example 2 : The two-point boundary value problem

$$u''(t) = u(t) + \sin(u(t)) \qquad (0 \le t \le 1), \tag{B.7}$$

$$u(0) = 0, \qquad u(1) = 1 \tag{B.8}$$

may be discretized [OrtR-69a] to yield $f(x) = 0$ where $f : R^n \to R^n$ is of the form (B.3) where $A \in M(R^n)$ is given by (B.4), $d : R^n \to R^n$ is defined by

$$d_i(x) = h^2(x_i + \sin(x_i)) \qquad (i = 1, \ldots, n), \tag{B.9}$$

where $h = \frac{1}{n+1}$, and $c \in R^n$ is given by

$$c_i = \begin{cases} 0 & (i = 1, \ldots, n-1) \\ -1 & (i = n) \end{cases} . \tag{B.10}$$

Example 2 is quoted by Alefeld and Platzoder [AleP-83a]. □

Example 3 : The two-point boundary value problem

$$u''(t) = \exp(u(t)) \qquad (0 \le t \le 1), \tag{B.11}$$

$$u(0) = 0, \qquad u(1) = 0 \tag{B.12}$$

may be discretized [Hen--62a] to yield $f(x) = 0$ where $f : R^n \to R^n$ is of the form

$$f(x) = Ax + Td(x) + c \tag{B.13}$$

in which $A \in M(R^n)$ is given by (B.4), $T \in M(R^n)$ is given by

$$T = \begin{pmatrix} \frac{1}{12} & \frac{10}{12} & & & \bigcirc \\ \frac{10}{12} & \frac{1}{12} & \frac{10}{12} & & \\ & \ddots & \ddots & \ddots & \\ & & \frac{10}{12} & \frac{1}{12} & \frac{10}{12} \\ \bigcirc & & & \frac{10}{12} & \frac{1}{12} \end{pmatrix}_{n \times n}, \tag{B.14}$$

$d : R^n \to R^n$ is defined by

$$d_i(x) = h^2 \exp(x_i) \qquad (i = 1, \ldots, n) \tag{B.15}$$

where $h = \frac{1}{n+1}$, and $c \in R^n$ is given by

$$c_i = \begin{cases} \frac{h^2}{12} & (i = 1) \\ 0 & (i = 2, \ldots, n-1) \\ \frac{h^2}{12} & (i = n) \end{cases}. \quad \square \tag{B.16}$$

Example 4 : The two-point boundary value problem

$$u''(t) = 2h^2(u(t) - t/2 + 1)^3 \qquad (0 \le t \le 1), \tag{B.17}$$

$$u(0) = 0, \qquad u(1) = 0 \tag{B.18}$$

may be discretized [Hen--62a] to yield $f(x) = 0$ where $f : R^n \to R^n$ is of the form (B.13), in which $A \in M(R^n)$ is given by (B.4), $T \in M(R^n)$ is given by (B.14), $d : R^n \to R^n$ is defined by

$$d_i(x) = 2h^2(x_i - t_i/2 + 1)^3 \qquad (i = 1, \ldots, n) \tag{B.19}$$

where $h = \frac{1}{n+1}$ and $t_i = ih \quad (i = 1, \ldots, n)$ and $c \in R^n$ is given by

$$c_i = \begin{cases} \frac{h^2}{16} & (i = 1) \\ 0 & (i = 2, \ldots, n-1) \\ \frac{9h^2}{16} & (i = n) \end{cases} . \quad \square \tag{B.20}$$

Example 5 : The two-point boundary value problem

$$u''(t) = \frac{1}{2}(u(t))^3 \qquad (0 \leq t \leq 1), \tag{B.21}$$

$$u(0) = 1, \qquad u(1) = 2 \tag{B.22}$$

may be discretized [Hen--62a] to yield $f(x) = 0$ where $f : R^n \to R^n$ is of the form (B.13) in which $A \in M(R^n)$ is given by (B.4), $T \in M(R^n)$ is given by (B.14), $d : R^n \to R^n$ is defined by

$$d_i(x) = \frac{h^2}{2} x_i^3 \qquad (i = 1, \ldots, n), \tag{B.23}$$

where $h = \frac{1}{n+1}$ and $c \in R^n$ is given by

$$c_i = \begin{cases} \frac{h^2}{24} - 1 & (i = 1) \\ 0 & (i = 2, \ldots, n-1) \\ \frac{h^2}{3} - 2 & (i = n) \end{cases} . \quad \square \tag{B.24}$$

Example 6 : The elliptic boundary value problem

$$\nabla^2 u(s,t) = (1 + u(s,t))^3 \qquad ((s,t) \in \Omega), \tag{B.25}$$

$$u(s,t) = 0 \qquad ((s,t) \in \partial\Omega), \tag{B.26}$$

where $\Omega = [0,1] \times [0,1]$ and $\partial\Omega$ is the boundary of Ω, may be discretized [OrtR-69a] to yield $f(x) = 0$ where $f : R^n \to R^n$ is of the form (B.3) in which $A \in M(R^n)$ is the block tridiagonal matrix

$$A = \begin{pmatrix} B & -I_m & & \bigcirc \\ -I_m & B & -I_m & \\ & \ddots & \ddots & \ddots \\ & & -I_m & B & -I_m \\ \bigcirc & & & -I_m & B \end{pmatrix}_{n \times n} \tag{B.27}$$

where $n = m^2$, I_m is the $m \times m$ identity matrix and

$$B = \begin{pmatrix} 4 & -1 & & \bigcirc \\ -1 & 4 & -1 & \\ & \ddots & \ddots & \ddots \\ & & -1 & 4 & -1 \\ \bigcirc & & & -1 & 4 \end{pmatrix}_{m \times m} , \tag{B.28}$$

$d : R^n \to R^n$ is defined by

$$d_i(x) = h^2(1 + x_i)^3 \qquad (i = 1, \ldots, n) \tag{B.29}$$

where $h = \frac{1}{m+1}$, and $c \in R^n$ is given by

$$c_i = 0 \qquad (i = 1, \ldots, n). \quad \square \tag{B.30}$$

Example 7 : The elliptic boundary value problem

$$\nabla^2 u(s,t) = (u(s,t))^2 \qquad ((s.t) \in \Omega) \tag{B.31}$$

$$u(s,o) = 2s^2 - s + 1 \qquad (0 \le s \le 1) \tag{B.32}$$

$$u(1,t) = 2 \qquad (0 \le t \le 1) \tag{B.33}$$

$$u(s,1) = 2 \qquad (0 \le s \le 1) \tag{B.34}$$

$$u(0,t) = 2t^2 - t + 1 \qquad (0 \le t \le 1) \tag{B.35}$$

where $\Omega = [0,1] \times [0,1]$ may be discretized [OrtR-69a] to yield $f(x) = 0$ where $f : R^n \to R^n$ is of the form (B.3), in which $A \in M(R^n)$ is given by (B.27) and (B.28), $d : R^n \to R^n$ is defined by

$$d_i(x) = h^2 x_i^2 \qquad (i = 1, \ldots, n) \tag{B.36}$$

where $h = \frac{1}{m+1}$ and $c \in R^n$ is given by $c_{lm+k} = \varphi_{kl}$ $(k, l = 1, \ldots, m)$ and

$$\varphi_{kl} = \begin{cases} -4h^2 + 2h - 2 & (k = 1,\ l = 1) \\ -2m^2h^2 + mh - 3 & (k = 1,\ l = m) \\ -2m^2h^2 + mh - 3 & (k = m,\ l = 1) \\ -4 & (k = m,\ l = m) \\ -2l^2h^2 + lh - 1 & (k = 1,\ l = 2, \ldots, m-1) \\ -2 & (k = m,\ l = 2, \ldots, m-1) \\ -2 & (k = 2, \ldots, m-1,\ l = m) \\ -2k^2h^2 + kh - 1 & (k = 2, \ldots, m-1,\ l = 1) \\ 0 & \text{otherwise} \end{cases} . \tag{B.37}$$

Example 7 is quoted by Rall [Ral--69a]. □

Example 8 : The function $f : R^n \to R^n$ defined by

$$f_i(x) = x_i(2 + 5x_i^2) + 1 - \sum_{i \in J_i} x_j(1 + x_j) \qquad (i = 1, \ldots, n) \tag{B.38}$$

where

$$J_i = \{j : j \neq i, \max(1, i-5) \leq j \leq \max(n, i+1)\} \qquad (B.39)$$

is the Broyden banded function. This example is quoted by Hansen and Greenberg [HanG-83a]. □

Example 9 : The system of equations $f(x) = 0$ where $f : R^n \to R^n$ is defined by

$$f_i(x) = -(n+1) + 2x_i + \sum_{j \neq i} x_j, \qquad (i = 1, \ldots, n-1),$$

$$f_n(x) = -1 + \prod_{j=1}^{n} x_j,$$

is described by Dennis and Brown [DenB-71a].

Example 10 : The system of equations $f(x) = 0$ where $f : R^n \to R^n$ is defined by

$$f_i(x) = \begin{cases} n(x_i - 1) & (1 = 1) \\ n(x_i - 1) - \sum_{j=1}^{i-1}(x_i^2 - 1) & (i = 2, \ldots, n) \end{cases}, \qquad (B.40)$$

is specially constructed to illustrate the effectiveness of the interval Brown algorithm described in Chapter 7. □

Appendix C — Description of the Pseudo-code

This section contains a brief description of the pseudo-code in which the *ALGLIB* package is written. The format for the pseudo-code is designed to be self-explanatory. The control structures used are based on the S-algol control structures since their function is obvious from their wording. Statements which are blocked together are enclosed between begin and end. A numbering scheme and indentation is also used to reflect the block structure of the algorithm. The numbering is also valuable since it is often useful to be able to refer to a specific line in a procedure.

C.1 Procedure declaration and calling

It is neccessary to have a well-defined format for declaring and calling procedures included in the pseudo-code. Otherwise recursive algorithms are difficult to express. Two kinds of procedure are used. The first does not return a specific value but has input, input/output and output parameters in much the same way as a FORTRAN SUBROUTINE or PASCAL procedure. The second returns a specific value but has only input parameters, and is similar to a FORTRAN FUNCTION or PASCAL function. The first type of procedure is declared using the syntax

procedure ⟨procedure name⟩ (⟨input parameters⟩ ;
⟨input/output parameters⟩ : ⟨output parameters⟩)
⟨block of statements⟩
return

where parameters of a given type are separated by commas. The block of statements making up the procedure body is terminated by the return statement. For example the procedure *example.proc* with one integer input parameter n , one real input/output parameter x , and two real vectors a and b as output parameters has the form

procedure *example.proc*$(n \in Z; x \in R : a, b \in R^n)$
⟨block of statements⟩
return

To invoke this type of procedure just use the name followed by an actual parameter list. The above procedure could be invoked by the statement

example.proc$(m; y : p, q)$

This call executes the procedure with the actual parameters m , y , p , q replacing the formal parameters n , x , a , b respectively. On return, the values of y , p and q could have been changed but the value of m could not.

The second type of procedure is defined using the syntax

procedure ⟨procedure name⟩ (⟨input parameters⟩ → ⟨output type⟩)
  ⟨block of statements⟩
return(⟨object returned⟩)

For example consider the procedure f , which has one real input parameter and which returns a real value.

procedure $f(x \in R \to R)$
  ⟨block of statements⟩
return(⟨real value to be returned⟩)

To call this type of procedure we could use the statement

Type	Notation	Vector Notation	Matrix Notations	
integer	Z	Z^n	$M(Z^m, Z^n)$	$M(Z^n)$
real	R	R^n	$M(R^m, R^n)$	$M(R^n)$
string	S	S^n	$M(S^m, S^n)$	$M(S^n)$
boolean	B	B^n	$M(B^m, B^n)$	$M(B^n)$
interval	$I(R)$	$I(R^n)$	$I(M(R^m, R^n))$	$I(M(R^n))$
pointer	P	P^n	$M(P^m, P^n)$	$M(P^n)$

Table C.1

$f.x := f(y)$

In the parameter list for both types of procedures, as well as the name of the parameter, a type is assigned to it. The notation used is as follows. There are six simple data types, namely integer, real, string, boolean, interval and pointer. Table C.1 summarizes the representation for these simple data types. The table also gives the notation by which vectors and matrices, whose elements are of any of the above types may be represented.

C.2 Components of vectors or matrices

Components of vectors are accessed by enclosing the index/indices of the required component in brackets after the vector's name or the matrix's name. Thus

$$vars(i) \qquad \text{and} \qquad A(i,j)$$

could be used to access the i^{th} element of the vector vars and the ij^{th} element of the matrix A .

C.3 Control Structures

Descriptions of the control structures are given below. In these descriptions ⟨statement⟩ denotes either a single statement, or a block of statements enclosed in a **begin**...**end** , and ⟨condition⟩ can be a simple condition of the form $a = b$, a compound condition of the form $a < b$ and $x = y$, or a boolean variable.

if...do

Syntax: **if** ⟨condition⟩ **do** ⟨statement⟩

Meaning: If ⟨condition⟩=true then execute ⟨statement⟩.

if...then...else

Syntax: **if** ⟨condition⟩ **then** ⟨statement1⟩ **else** ⟨statement2⟩

Meaning: if ⟨condition⟩=true then execute ⟨statement1⟩; Otherwise execute ⟨statement2⟩.

case

Syntax:

case

⟨condition1⟩	:	⟨statement1⟩
⋮		⋮
⟨condition n⟩	:	⟨statement n⟩
default	:	⟨statement n + 1⟩

Meaning: For $i = 1, \ldots, n$, if ⟨condition i⟩=true then execute ⟨statement i⟩ only. However if ⟨condition i⟩=false ($i = 1, \ldots, n$) then execute ⟨statement n + 1⟩ only.

for...to...do

Syntax: for ⟨integer variable⟩ = ⟨integer value 1⟩ to ⟨integer value 2⟩ do ⟨statement⟩

Meaning: If ⟨integer value 1⟩ is greater than ⟨integer value 2⟩ then do not execute ⟨statement⟩. Otherwise give the value ⟨integer value 1⟩ to ⟨integer variable⟩ and execute ⟨statement⟩ repeatedly, incrementing ⟨integer variable⟩ by unity each time ⟨statement⟩ is executed, until ⟨integer variable⟩ is greater than ⟨integer value 2⟩.

while...do

Syntax: while ⟨condition⟩ do ⟨statement⟩

Meaning: While ⟨condition⟩=true repeatedly execute ⟨statement⟩.

repeat...while

Syntax: repeat ⟨statement⟩ while ⟨condition⟩

Meaning: Execute ⟨statement⟩ repeatedly until ⟨condition⟩=false.

repeat...while...do

Syntax: repeat ⟨statement1⟩ while ⟨condition⟩ do ⟨statement2⟩

Meaning: Repeatedly { execute ⟨statement1⟩ and if ⟨condition⟩=true then execute ⟨statement2⟩ } until ⟨condition⟩=false.

Assignment

Syntax: ⟨variable⟩ := ⟨value⟩

Meaning: Assign ⟨value⟩ to ⟨variable⟩.

Comment

Syntax: ! ⟨text⟩

Meaning: Treat ⟨text⟩ as comment.

References

[Ale--72a] G. Alefeld, *Über die Existenz einer eindeutigen Lösung bei einer Klasse nichtlinearer Gleichungssysteme und deren Berechnung mit Iterationsverfahren*, Aplikace Matematiky 17: 267–294 (1972).

[Ale--77a] G. Alefeld, *Das symmetrische Einzelschrittverfahren bei linearen Gleichungen mit Intervallen als Koeffizienten*, Computing 18: 329–340 (1977).

[AleH-72a] G. Alefeld and J. Herzberger, *Nullstelleneinschliessung mit dem Newton-Verfahren ohne Invertierung von Intervallmatrizen*, Numer. Math. 19: 56–64 (1972).

[AleH-74a] G. Alefeld and J. Herzberger, *Einfuhrung in die Intervallrechnung*, Bibliographisches Institut, Mannheim, Vienna, Zurich, 1974.

[AleH-83a] G. Alefeld and J. Herzberger, *Introduction to Interval Computations*, Academic Press, New York, 1983.

[AleP-83a] G. Alefeld and L. Platzöder, *A quadratically convergent Krawczyk-like algorithm*, SIAM J. Numer. Anal. 20: 210–219 (1983).

[BaiC---a] P.J. Bailey, A.J. Cole and R. Morrison, *Triplex User Manual CS/82/5*, University of St. Andrews Department of Computational Science, North Haugh, St. Andrews, Fife, KY16 9SX, Scotland.

[Bog--77a] R.A. Bogen *et al.*, *MACSYMA Reference Manual*, MIT Laboratory for Computer Science, Cambridge, Mass., 1977.

[Bor--68a] D.G. Borrow, *Symbol Manipulation Languages and Techniques*, North Holland Publishing Company, Amsterdam, 1968.

[Bor--79a] R. Bornat, *Understanding and Writing Compilers*, Macmillan Press Ltd, London, 1979.

[Bre--73a] R.P. Brent, *Some efficient algorithms for solving systems of nonlinear algebraic equations*, SIAM J. Numer. Anal. 10: 327–344 (1973).

[Bro--12a] L.E.J. Brouwer, *Uber die Abbildung von Mannigfaltigkeiten*, Math. Ann. 71: 97–115 (1912).

[Bro--66a] K.M. Brown, *A quadratically convergent method for solving simultaneous non-linear equations*, PhD Thesis, Purdue University, 1966.

[Bro--67a] K.M. Brown, *Solution of simultaneous non-linear equations*, Comm. A.C.M. 10: 728–729 (1967).

[BroC-67a] K.M. Brown and S.D. Conte, *The solution of simultaneous nonlinear equations*, in Proc. 22nd Nat. Conf., A.C.M., 1967.

[ColM-82a] A.J. Cole and R. Morrison, *An Introduction to Programming with S-algol*, Cambridge University Press, Cambridge, 1982.

[ColM-82b] A.J. Cole and R. Morrison, *Triplex: A system for interval arithmetic*, Software—Practice and Experience 12: 341–350 (1982).

[DavM-81a] A.J.T. Davie and R. Morrison, *Recursive Descent Compiling*, Ellis Horwood Ltd, Chichester, 1981.

[DenB-71a] J.E. Dennis and K.M. Brown, *On the second order convergence of Brown's derivative-free method for solving simultaneous nonlinear equations*, Research Report Number 71-7, Department of Computer Science, Yale University, Connecticut, 1971.

[DenS-83a] J.E. Dennis and R.B. Schnabel, *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*, Prentice-Hall, Inc., Eaglewood Cliffs, 1983.

[Han--68a] E.R. Hansen, *On solving systems of equations using interval arithmetic*, Math. Comp. 22: 374–384 (1968).

[Han--69a] E. Hansen, *On linear algebraic equations with interval coefficients*, in Topics in Interval Analysis, Oxford University Press, London and New York, 1969.

[Han--78a] E.R. Hansen, *A globally convergent interval method for computing and bounding real roots*, BIT 18: 415–424 (1978).

[HanG-83a] E.R. Hansen and R.I. Greenberg, *An interval Newton method*, Appl. Math. Comput. 12: 89–98 (1983).

[HanS-81a] E.R. Hansen and S. Sengupta, *Bounding solutions of systems of equations using interval analysis*, BIT 21: 203–211 (1981).

[Hea--73a] A. Hearn, *REDUCE 2 User's Manual*, University of Utah, Utah, 1973.

[Heb--74a] M. Hebgen, *Eine scaling-invariante Pivotsuche fur Intervallmatrizen*, Computing 12: 107–115 (1974).

[Hen--62a] P. Henrici, *Discrete Variable Methods in Ordinary Differential Equations*, John Wiley & Sons Inc, New York, London, 1962.

[How--79a] J.C. Howard, *Practical Applications of Symbolic Computation*, IPC Science and Technology Press, Guildford, 1979.

[Jon--78a] S.T. Jones, *Searching for solutions of finite nonlinear systems—An interval approach*, PhD Thesis, Univ. of Wisconsin-Madison, Wisconsin, 1978.

[Jon--80a] S.T. Jones, *Locating safe starting regions for iterative methods: A heuristic algorithm*, in Interval Mathematics 1980, (K.L.E. Nickel, Ed.), Academic Press, New York, 1980.

[Kel--68a] H.B. Keller, *Numerical Methods for Two-point Boundary-value Problems*, Blaisdell Publishing Company, Waltham, Toronto, London, 1968.

[Kra--69a] R. Krawczyk, *Newton-Algorithmen zur Bestimmung von Nullstellen mit Fehlerschranken*, Computing 4: 187–201 (1969).

[Kup--67a] I.B. Kupermann, *Approximate linear algebraic equations and rounding error estimation*, PhD Thesis, Univ. of Witwatersand, Johannesburg, 1967.

[Mad--73a] K. Madsen, *On the solution of nonlinear equations in interval arithmetic*, BIT 13: 428–433 (1973).

[McC--83a] G.P. McCormick, *Nonlinear Programming — Theory, Algorithms, and Applications*, John Wiley and Sons, New York, Chichester, Brisbane, Toronto, Singapore, 1983.

[Mon--84a] M. Monsi, *Private Communication*, 1984.

[Mon--85a] M. Monsi, *Private Communication*, 1985.

[Mon--73a] W. Monch, *Einschliessung von positiven Inversen*, Z. Angew. Math. Mech. 53: 207–208 (1973).

[Moo--62a] R.E. Moore, *Interval arithmetic and automatic error analysis in digital computing*, PhD Thesis, Stanford University, 1962.

[Moo--66a] R.E. Moore, *Interval Analysis*, Prentice-Hall Inc., Englewood Cliffs, New Jersey, 1966.

[Moo--77a] R.E. Moore, *A test for existence of solutions to nonlinear systems*, SIAM J. Numer. Anal. 14: 611–615 (1977).

[Moo--78a] R.E. Moore, *A computational test for convergence of iterative methods for nonlinear systems*, SIAM J. Numer. Anal. 15: 1194–1196 (1978).

[Moo--78b] R.E. Moore, *Bounding sets in function spaces with application to nonlinear operator equations*, SIAM Review 20: 492–512 (1978).

[Moo--79a] R.E. Moore, *Methods and Applications of Interval Analysis*, SIAM Publications, Philadelphia, 1979.

[Moo--80a] R.E. Moore, *Interval methods for nonlinear systems*, Computing Supplementum 2: 113–120 (1980).

[MooJ-77a] R.E. Moore and S.T. Jones, *Safe starting regions for iterative methods*, SIAM J. Numer. Anal. 14: 1051–1065 (1977).

[MooQ-82a] R.E. Moore and L. Qi, *A successive interval test for nonlinear systems*, SIAM J. Numer. Anal. 19: 845–850 (1982).

[MorC-83a] R. Morrison, A.J. Cole, P.J. Bailey, M.A. Wolfe and J.M. Shearer, *Experience in using a high level language which supports interval arithmetic*, read at ARITH6, the sixth symposium on computer arithmetic, Aarhus, Denmark, 1983.

[Nic--71a] K.L.E. Nickel, *On the Newton method in interval analysis*, University of Wisconsin MRC Technical Summary Report #1136, 1971.

[OrtR-69a] J.M. Ortega and W.C. Rheinboldt, *Iterative Solution of Nonlinear Equations in Several Variables*, Academic Press, New York, London, 1969.

[Pug--72a] R.E. Pugh, *A language for nonlinear programming problems*, Math. Programming 2: 176–206 (1972).

[Qi---80a] L. Qi, *A generalization of the Krawczyk-Moore algorithm*, in Interval Mathematics 1980, (K.L.E. Nickel, Ed.), Academic Press, New York, 1980.

[Qi---82a] L. Qi, *A note on the Moore test for nonlinear systems*, SIAM J. Numer. Anal. 19: 851–857 (1982).

[Ral--69a] L.B. Rall, *Computational Solution of Nonlinear Operator Equations*, Wiley and Sons Inc, New York, 1969.

[Ral--80a] L.B. Rall, *A comparison of the existence theorems of Kantorovich and Moore*, SIAM J. Numer. Anal. 17: 148–161 (1980).

[Ral--80b] L.B. Rall, *Applications of software for automatic differentiation in numerical computation*, Computing Suppl. 2: 141–156 (1980).

[Rhe--74a] W.C. Rheinboldt, *Methods for Solving Systems of Nonlinear Equations. Regional Conference Series in Applied Mathematics, 14*, SIAM Publications, Philadelphia, 1974.

[Sis--82a] F.S. Sisser, *Computer-generated interval extensions of factorable functions and their derivatives*, Intern. J. Computer Math. 10: 327–336 (1982).

[Sis--82b] F.S. Sisser, *Inverting an interval Hessian of a factorable function*, Computing 29: 63–72 (1982).

[Sis--82c] F.S. Sisser, *A modified Newton's method for minimizing factorable functions*, Journal of Optimization Theory and Applications 38: 461–482 (1982).

[Var--62a] R. Varga, *Matrix Iterative Analysis*, Prentice Hall Inc., Englewood Cliffs, New Jersey, 1962.

[Wat--83a] S. Watanabe, *Hybrid manipulations for the solution of systems of nonlinear algebraic equations*, RIMS, Kyoto University 19: 367–395 (1983).

[Wol--80a] M.A. Wolfe, *A modification of Krawczyk's algorithm*, SIAM J. Numer. Anal. 17: 376–379 (1980).